Real-Time Power System Topology Monitoring Supported by Synchrophasor Measurements

by

Trevor Nelson Werho

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved November 2015 by the
Graduate Supervisory Committee:

Vijay Vittal, Chair
Gerald Heydt
Kory Hedman
George Karady

ARIZONA STATE UNIVERSITY

December 2015

ABSTRACT

This dissertation introduces a real-time topology monitoring scheme for power systems intended to provide enhanced situational awareness during major system disturbances. The topology monitoring scheme requires accurate real-time topology information to be effective. This scheme is supported by advances in transmission line outage detection based on data-mining phasor measurement unit (PMU) measurements.

A network flow analysis scheme is proposed to track changes in user defined minimal cut sets within the system. This work introduces a new algorithm used to update a previous network flow solution after the loss of a single system branch. The proposed new algorithm provides a significantly decreased solution time that is desired in a real-time environment. This method of topology monitoring can provide system operators with visual indications of potential problems in the system caused by changes in topology.

This work also presents a method of determining all singleton cut sets within a given network topology called the one line remaining (OLR) algorithm. During operation, if a singleton cut set exists, then the system cannot withstand the loss of any one line and still remain connected. The OLR algorithm activates after the loss of a transmission line and determines if any singleton cut sets were created. These cut sets are found using properties of power transfer distribution factors and minimal cut sets.

The topology analysis algorithms proposed in this work are supported by line outage detection using PMU measurements aimed at providing accurate real-time topology information. This process uses a decision tree (DT) based data-mining approach to characterize a lost tie line in simulation. The trained DT is then used to analyze PMU meas-

urements to detect line outages. The trained decision tree was applied to real PMU measurements to detect the loss of a 500 kV line and had no misclassifications.

The work presented has the objective of enhancing situational awareness during significant system disturbances in real time. This dissertation presents all parts of the proposed topology monitoring scheme and justifies and validates the methodology using a real system event.

ACKNOWLEDGMENTS

I would like to thank Dr. Vittal for all his help and support over the course of this project. I would like to thank the members of my committee for their suggestions and contributions that helped improve the quality of this work. I would also like to thank my parents for always being supportive while I pursue my educational goals. Finally, I would like to thank the PSERC industry sponsors that made this project possible.

TABLE OF CONTENTS

Page

LIST OF TABLES vi

LIST OF FIGURES vii

NOMENCLATURE ix

CHAPTER

1. INTRODUCTION 1

1.1 Introduction to the Research Focus 1

1.2 Phasor Measurement Units 5

1.3 Entergy System and Gustav Event Test Case 5

1.4 Network Flow Algorithms 7

1.5 The DC Power Flow Model 13

1.6 CART Algorithm and Database 15

1.7 Literature Review 18

1.7.1 Network Flows 18

1.7.2 Critical Line Identification 22

1.7.3 PMU Line Outage Detection 24

1.7.4 Applications of Decision Trees 28

1.8 Objectives 29

1.9 Organization of This Report 31

CHAPTER Page

2. TOPOLOGY ANALYSIS 32

2.1 Network Flow Analysis 32

2.1.1 Algorithm to Update a Network Flow Solution 33

2.1.2 Network Flows Applied to the Gustav Test Case 39

2.1.3 Advantages of Using Network Flow Monitoring 42

2.2 One Line Remaining Algorithm 49

2.2.1 Algorithm Explanation 51

2.2.2 OLR Algorithm Validation 56

3. LINE OUTAGE DETECTION 63

3.1 PMU-Based Line Outage Detection 63

3.1.1 Hurricane Isaac Simulations 63

3.1.2 Feature Selection 67

3.1.3 Decision Tree Training and Testing 68

4. CONCLUSIONS 72

REFERENCES 76

APPENDIX

A MATLAB IMPLEMENTATION OF NETWORK FLOW ALGORITHMS 81

B MATLAB IMPLEMENATION OF THE OLR ALGORITHM 89

LIST OF TABLES

Table Page

1.1 Time Instant of Tie Line Outages During Hurricane Gustav 8

1.2 Example Admittance Matrix 13

1.3 Example CART Database 19

2.1 Network Flow Solutions Following the Loss of Each Transmission Line During the Gustav Event 41

2.2 DC Power Flow Solution of the 4 Bus System Example 53

2.3 OLR Solutions for Several Outages in the 16 Bus System 58

2.4 OLR Solutions for a Series of Outages in the IEEE 118 Bus System 58

2.5 OLR Algorithm Solutions for the Gustav Event 62

3.1 DT Training Results 70

LIST OF FIGURES

Figure Page

1.1 The Entergy System and the Five Major Operating Areas [13] 6

1.2 A Six Vertex Directed Graph..... 10

1.3 The Edge Flows After the First Iteration of Edmonds and Karp..... 10

1.4 The Edge Flows After the Second Iteration of Edmonds and Karp 11

1.5 The Maximum Flow After the Third Iteration of Edmonds and Karp 11

1.6 The Directed Representation of a Bidirectional Graph..... 12

1.7 Example CART Decision Tree 20

1.8 Filtered and Unfiltered PMU Data During a 500 kV Line Outage [6] 27

1.9 Voltage Phase Angle PMU Data Showing a 500 kV Line Outage During the Gustav Event 27

2.1 The Maximal Flow in a Network with All Edge Capacities of One..... 34

2.2 The Updated Solution for the Loss of the Type One Edge 2 to 6..... 37

2.3 The Updated Solution for the Loss of the Type Two Edge 1 to 2 39

2.4 The Maximum Flow to Selected Sinks Within the Entergy System Prior to the Event. 43

2.5 The Maximum Flows to Selected Sinks Within the Entergy System at Approximately 1:30 PM During Hurricane Gustav 44

2.6 The Maximum Flows to Selected Sinks Within the Entergy System at Approximately 2:10 PM During Hurricane Gustav 45

2.6 Example Power System with Two Tie Lines..... 50

2.7 Simple 4 Bus Power Network..... 51

Figure Page

2.8 Admittance Matrix of the Network Shown in Fig 2.7 52

2.9 Admittance Matrix of the Network Shown in Fig. 2.7 After Suffering the Loss of Branch 1 to 2 52

2.10 16 Bus Power System Network 57

2.11 IEEE 118 Bus Power System 60

2.12 PSS/E Drawing of the Entergy System in Southeast Louisiana. The PMUs Within the Island and the Endpoints of the Final Two Tie Lines are All Labeled. 61

3.1 Simulated Voltage Phase Angle Difference Between Buses 75400 and 52905 65

3.2 Simulated Voltage Magnitude at Waterford Inside the Island Area 66

3.3 Voltage Phase Angle Difference Between the Inside and Outside of the Island Area Measured by PMUs During the Loss of 55026-63411 (500 kV) 66

3.4 DT Trained with the Learning Dataset 69

3.5 DT Applied to PMU Measurements of the Real 500 kV Line Outage 70

NOMENCLATURE

B	A susceptance
BFS	Breadth first-search
$C(i \mid j)$	The cost of misclassifying a class j as a class i
CA	Critical attribute in a decision tree or Contingency Analysis
CAISO	California Independent Transmission System Operator
CART	Classification and Regression Trees
Con Edison	Consolidated Edison Company
CSR	Critical splitting rule in a decision tree
DC	Direct current
d_{lk}	The line outage distribution factor looking at line l after an outage of line k
DT	Decision tree
E	Voltage magnitude or the number of edges in a graph
f_k^0	The original power flow on line k
f_l	The power flow on line l
G	A conductance
GHz	Gigahertz
GPS	Global Positioning System
h, l	Line labels
H_a	Matrix of power transfer distribution factors
IID	Imperial Irrigation District

$i(t)$	The Gini impurity at node t
i, j, and k	Indices
kV	Kilovolts
L	A lower left triangular matrix that is part of a factored admittance matrix
LODF	Line outage distribution factor
Max1	A feature used in decision tree training
Max2	A feature used in decision tree training
Min1	A feature used in decision tree training
Min2	A feature used in decision tree training
MISO	Midcontinent Independent System Operator
MW	Megawatts
n	The number of buses in a system or a specific angle measurement in edge detection
N	The number of buses in a system
$N+$, $N-$	Two disjoint sets
O	In the order of
OLR	One line remaining
$p(i \mid t)$	The probability of a case being class i given that it falls into node t
$p(t)$	The probability of a case being in node t
P_i	The real power injection at bus i
P_{ik}	The real power flow from bus i to bus k

P_l^{ij}	A power transfer distribution factor, the change in flow on line l for a power transaction from point i to point j
p_l	The probability of sending a case to the left child node
PMU	Phasor measurement unit
PSS/E	Power systems analysis software
PTDF	Power transfer distribution factor
Q_i	The reactive power injection at bus i
r_i	The number of non-zeros to the right of the diagonal in the U matrix
R	A resistance
s	The number of sinks within the system
SCADA	Supervisory control and data acquisition
Slope 10	A feature used in decision tree training
Slope 5	A feature used in decision tree training
Slope 50	A feature used in decision tree training
TVA	Tennessee Valley Authority
U	An upper right triangular matrix that is part of a factored admittance matrix
V	The number of vertices in a graph
X	A reactance or the number of buses or the number of buses found by the OLR algorithm
x_{ik}	The reactance between bus i and bus k

Y	The admittance matrix
Y_{ij}	The i^{th}, j^{th} element in the admittance matrix
Δf_l	The change in power flow on line l
$\Delta i(s,t)$	The Gini impurity improvement
θ	A voltage phase angle
τ	The threshold used to identify events in edge detection

1. INTRODUCTION

1.1 Introduction to the Research Focus

Under normal operation, different areas of the power system are interconnected together and operated synchronously to provide enhanced reliability. However, under rare circumstances, an area of the system can become unintentionally separated from the rest of the grid, forming an electrical island. If the isolated portion of the system has a large imbalance between load and generation, the area can experience large fluctuations in system frequency and bus voltages. This can lead to some loss of load due to automatic protection actions or even a complete blackout of the area. These system events within modern power systems are rare occurrences. However, when these events occur, they often involve the loss of many components over a significant amount of time, which leads to a rapid and complete loss of load. For example, the 1977 Consolidated Edison Company blackout consisted of 11 transmission lines or transformers outages over the course of 52 minutes [1]. These line outages represented the loss of critical interties that connected the Consolidated Edison Company (Con Edison) to neighboring systems. One of the causes of the failure was credited to "Failure to recognize that a critical interconnection to the West (Y84) was effectively unavailable" [1]. Knowing the availability of an intertie can be difficult. An intertie is not necessarily a single transmission line whose status can be monitored. Rather, an intertie can be a portion of the network. This work proposes the application of topology based analyses during real-time power system operation, assisted by advances in synchrophasor measurement line outage verification, to provide better network visualization and awareness.

Two different topology based analysis methods are proposed in this work, which utilize graph theory and numerical methods. A graph theory based method is introduced using maximum flow network flow algorithms to determine cut sets to different locations within the system. In graph theory, a network flow can determine the maximum flow from a source to a sink within a directed graph [2]. Each branch in the network has a specific capacity. Normally the capacity of each branch in the network is associated with some quantity, such as amount of information in an information network or current in an electric power network. However, the method proposed in this work uses network flow algorithms where each branch in the network has a capacity of exactly one unit of flow. This will result in a network flow solution where the maximum flow is equal to the minimum number of branches needed to be lost in order to guarantee disconnecting the source from the sink [2]. As outages occur within the system, the value of maximum flow is a strong indicator to system operators as to the availability of an intertie. The value of using the proposed method is illustrated by discussing several major system events including a detailed example of the 2008 island formation that occurred in the Entergy power system. This island formation was caused by hurricane Gustav after 14 transmissions lines were lost over the course of 8 hours [3]. The method was applied to a recreation of the 2008 event using a 20,000 bus model of the Entergy system.

The proposed network flow method utilizes well-known graph theory network flow algorithms [4, 5], but also introduces a new algorithm for updating an old network flow solution. This new algorithm takes advantage of the fact that the topology of a power system changes very slowly, often times changing by only the status of a single line. When only a single line is removed from the system, a new network flow solution is not

needed. Instead, the old solution can be modified for the slight changes in system topology for a solution time reduction that is specific to this application.

The network flow analysis method provides important information about the topology of the system following transmission line outages. This information is intended to inform operators when the system topology changes and help visualize and understand the effects of the change. However, the only time corrective actions are required based solely on the network flow solutions are when a group of buses could be disconnected from the grid by the loss of only a single line. To assist with this situation, the second topology based analysis method proposed in this work utilizes numerical methods and is called the one line remaining (OLR) algorithm. The OLR algorithm is focused on identifying potential island formation situations based on the criteria that a number of buses are now connected to the rest of the system by only a single branch. These important branches are referred to as critical lines or critical branches in this work. The OLR algorithm utilizes a special DC power flow model of the system to obtain information about any and all critical lines within the system. The OLR algorithm was also applied to the Entergy Gustav island event to illustrate the information that would be available to system operators if such a system had been in place.

Both topology based analysis methods presented here rely on the availability of accurate system topology information in order to be effective. Telemetry data from the supervisory control and data acquisition system (SCADA) cannot always be relied upon. To aid with this issue, this work presents advances in line outage verification by using phasor measurement unit (PMU) measurements. Line outage verification is achieved by detecting when a transmission line has been lost by the constant monitoring of PMU

measurements within the system. Once a transmission line outage is *detected*, the interval of measurements just following the outage is examined and compared to simulated outages within a DC power flow model of the system. The comparison between PMU measurements and simulation is used to *match* the observed outage to a specific line within the system. This process was proposed in [6]. The study done in [6] uses an edge detection method to detect when a line is lost within the system. The edge detection method used in [6] used a small threshold to determine when an event had begun within the PMU measurements. This approach may not work in all cases. For example, the PMU measurements recorded during the Entergy island event in 2008 contained much larger variations in the data. Therefore, a more robust method of line out detection is presented in this work. The accurate detection of line outages is achieved by data-mining PMU measurements using the program CART (classification and regression trees). The program CART is designed to use a database of past measurements to train a decision tree (DT) that can be used to classify future inputs. In this work, line outages are conducted in simulation to create a sufficient CART database. The database is then used by CART to train a DT. Finally, the DT can be used with PMU measurements to detect line outages within the system. Together, these power system topology analysis methods, supported by advances in PMU based line outage verification, amount to an improvement in visualization and system awareness for operators during topology based system disturbances.

1.2 Phasor Measurement Units

Since their introduction in the 1980s, synchronized phasor measurement units have become a mature technology with many differently applications being developed around the world [7]. A PMU is a device that is located at a substation. The PMU takes instantaneous measurements at a particular bus in the system and returns an approximate phasor quantity. PMU measurements have magnitude accuracy that is better than 0.1% [8]. Each PMU is equipped with a Global Positioning System (GPS) receiver, which receives a signal from the GPS system containing the year, day, hour, minute, and second [9]. The signal from the GPS system is received once every second. The PMU then internally divides down this signal to allow sampling rates of 12, 30, 48, or 60 samples per second. The PMUs used in this work take 30 samples every second. The PMU uses the GPS time signal to accurately time stamp each measurement before it is sent off to a phasor data concentrator or the control center. Each PMU measurement is time stamped with precision better than 1 microsecond [8]. Once all the measurements are received by the control center, the measurements from different PMUs are matched together via time stamps to allow a clear picture of the system at a specific time regardless of the distance separating the measurement locations. This ability allows PMU measurements to have a major advantage over traditional SCADA monitoring.

1.3 Entergy System and Gustav Event Test Case

Throughout this work, the Gustav event is used as a test case to validate as well as display the benefits of the proposed approach within an actual power system disturbance. The Entergy power system serves 2.8 million customers in Arkansas, Louisiana, Missis-

sippi, and Texas. The system consists of over 1,500 substations with over 15,500 miles of lines at voltages of 69 kilovolts (kV) and above. The system has a total generating capacity of around 30,000 megawatts (MW) with 10,000 MW coming from nuclear sources. The system is part of the Eastern Interconnection in North America [10]. When this study was conducted, the system included 19 PMUs that are located at important buses throughout the system [11]. The Entergy system has 5 major operating areas: WOTAB, Amite South, Central, Sheridan North, and Dell. The locations of the different areas with the Entergy system can be seen in Fig. 1.1.

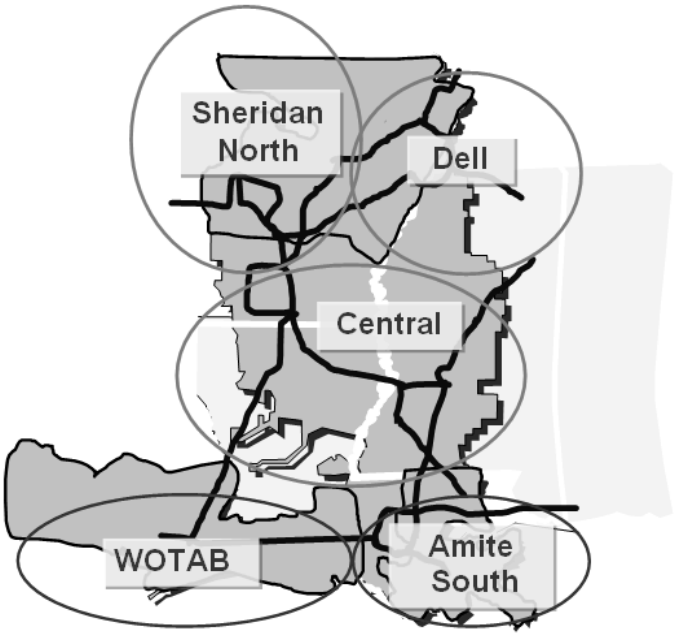

Fig. 1.1 The Entergy System and the Five Major Operating Areas [13]

On September 1, 2008 at 9:30 AM hurricane Gustav made landfall close to New Orleans. Over the course of several hours the Entergy system lost 13 tie lines that interconnected the Baton Rouge and New Orleans area to the rest of the grid. At 2:49 PM the 14th and final tie line was tripped that resulted in the formation of an electrical island containing most of Baton Rouge and New Orleans within the Amite South operating area.

The time instant of each tie line outage can be seen in Table 1.1. The outages shown in Table 1.1 were not the only components lost during the hurricane. Hurricane Gustav caused an outage of 241 transmission lines, 354 substations, and left 4,349 transformers damaged or destroyed [12]. During this destructive storm, system operators were not aware of the threat of islanding. Following the final tie line trip, system operators were first altered to the island formation by diverging frequencies measured at different PMU locations. The island was only identified 20 minutes after the island formed [12].

The island formed was composed of 139 buses. The island contained 3 generators and 2 PMUs. The three generating units within the island were all fossil fuel units producing 349 MW. The three generators had a combined maximum output of over 1500 MW. All of the nuclear generators in the system had been turned off in preparation for the storm. The total load in the island was approximately 249 MW. At the time just before the island formed the area was exporting approximately 100 MW. Following the island formation, the 3 generators within the area were able to regulate the frequency. The area continued to operate independently from the grid despite no operator actions being taken in preparation of the island formation.

1.4 Network Flow Algorithms

One of the network analysis methods proposed in this work utilizes graph theory network flow algorithms. In graph theory, a graph is composed of vertices and edges. Vertices act as nodes within the graph while edges connect vertices together [2]. If a power system is described as a set of buses connected together by transmission lines, then

buses in the system would become vertices and the transmission lines would become edges.

Table 1.1 Time Instant of Tie Line Outages During Hurricane Gustav

Tie Line Lost	Time of Outage
1st	8:52 AM
2nd	9:21 AM
3rd	10:19 AM
4th	11:40 AM
5th	11:59 AM
6th	12:15 PM
7th	12:28 PM
8th	12:45 PM
9th	1:14 PM
10th	1:16 PM
11th	1:22 PM
12th	1:28 PM
13th	2:08 PM

A network flow algorithm can determine the maximum flow that can be delivered from a source to a sink within a directed graph where each edge has a specific flow capacity. A flow represents some physical quantity being transported such as information in an information network or current in an electrical network. A directed graph is a graph composed of directed edges. Directed edges can only allow flow in one direction. The capacity of the edge in the opposite direction is zero [2]. One of the earliest network flow algorithms was created by Edmonds and Karp in 1969 [4]. Consider the directed graph depicted in Fig. 1.2. There is a fraction associated with each edge in the graph. The numerator indicates the present amount of flow crossing the edge while the denominator represents the maximum possible flow allowed. The Edmonds and Karp algorithm can be

used to find the maximum flow from vertex 1 to vertex 6 in this graph. A network flow solution must satisfy specific criteria in order to be correct [5].

1. For every edge, the flow across an edge cannot exceed the capacity of that edge.
2. For every vertex other than the source and the sink, the sum of the flows entering a vertex must equal the flow exiting that vertex.
3. The flow from source to sink must be maximal. The flow is maximal, if and only if, the graph contains no more augmenting paths between the source and the sink.

The Edmonds and Karp algorithm finds a solution by finding augmenting paths between the source and the sink. A path is a set of vertices and edges in a series where no vertex or edge is repeated. An augmenting path is a path where each edge along the path has remaining capacity. These paths are found using the breadth first-search (BFS) path finding algorithm [14]. Once such a path is found, flow is added to each edge along the path and the process starts again. When no more augmenting paths are found the algorithm terminates and a solution has been reached. The Edmonds and Karp algorithm requires three iterations to find a solution for the graph in Fig. 1.2. The results after each iteration of the algorithm can be seen in Figs. 1.3-1.5. Notice that the graph in Fig. 1.5 satisfies all of the solution criteria discussed previously and finds the network has a maximum flow of 6.

The Edmonds and Karp algorithm executes in $O(E^2V)$ time, where E is the number of edges in the graph and V is the number of vertices in the graph. More sophisticated algorithms have achieved reduced orders of complexity, such as Dinic's algorithm $O(EV^2)$, Karzanov's algorithm $O(V^3)$, and Goldberg and Tarjan $O(EV \log(V^2/E))$ [4]. The application proposed in this work can utilize any of these algorithms.

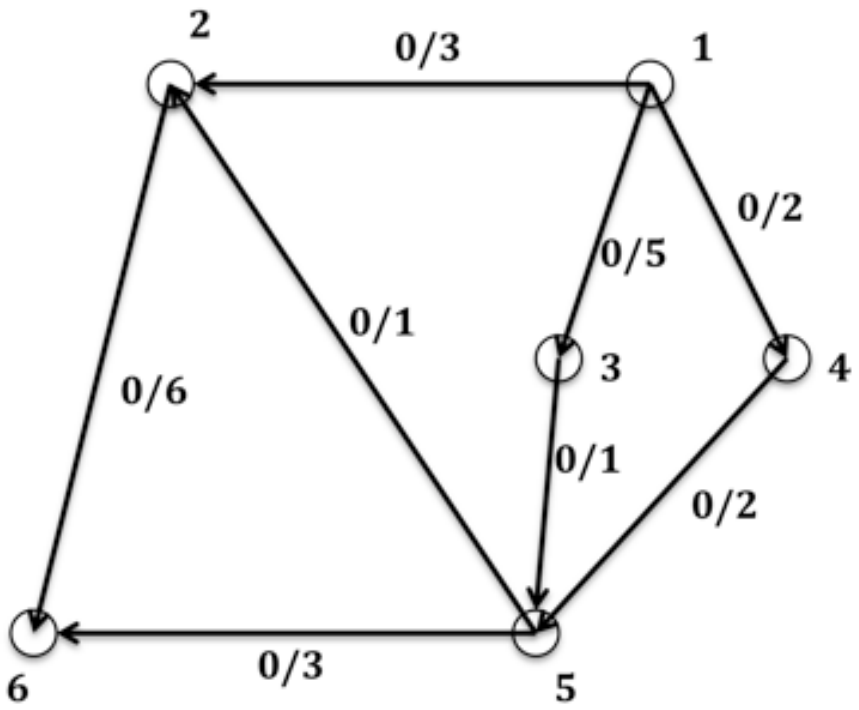

Fig. 1.2 A Six Vertex Directed Graph

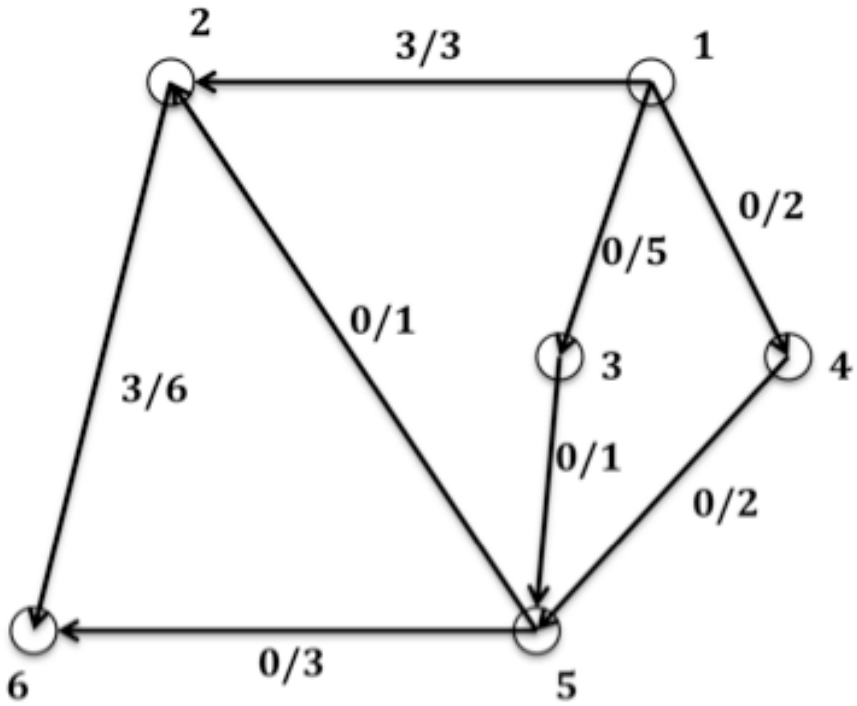

Fig. 1.3 The Edge Flows After the First Iteration of Edmonds and Karp

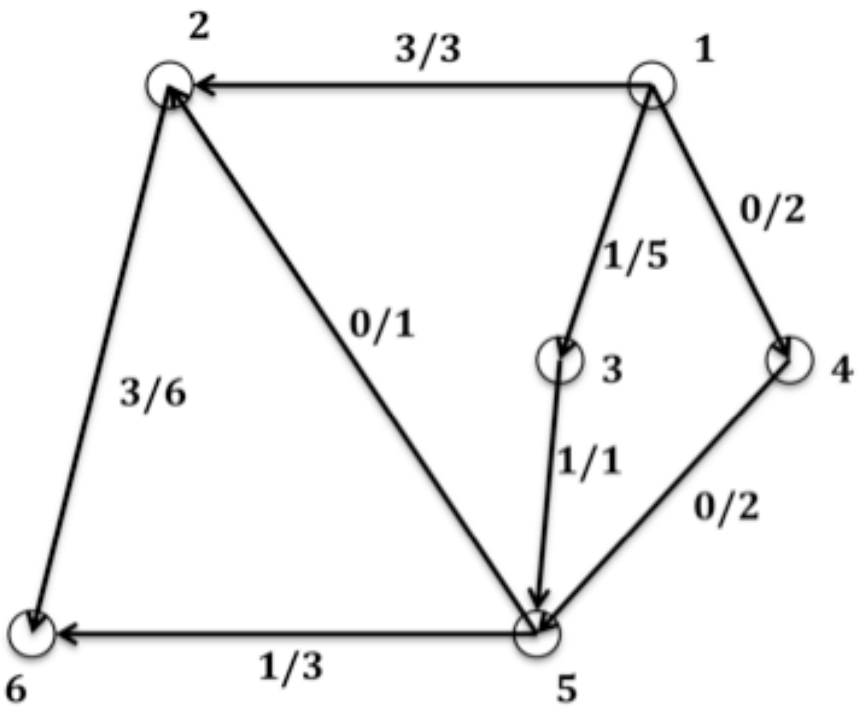

Fig. 1.4 The Edge Flows After the Second Iteration of Edmonds and Karp

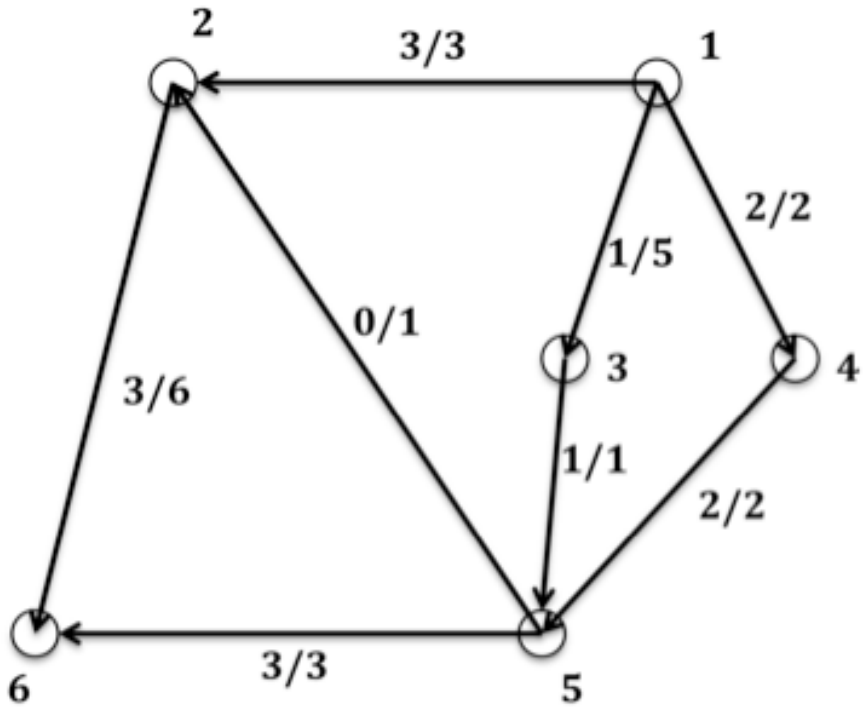

Fig. 1.5 The Maximum Flow After the Third Iteration of Edmonds and Karp

A power system is not directed in nature because power can flow in either direction across a transmission line. Representing each transmission line as two directed edges with equal capacity and opposite directions could solve this problem. Consider the directed representation of a bidirectional graph shown in Fig. 1.6. The Edmonds and Karp algorithm can also be applied to such a representation. To prevent unnecessary complication when using this representation, anytime flow is added to an edge, the algorithm checks the value of flow along the edge in the opposite direction. If flow is also on the opposing edge, then this flow is first reduced before new flow is added to the graph. This process is handled "behind the scenes" within the algorithm program. The remainder of this work will only discuss bidirectional graphs for simplicity.

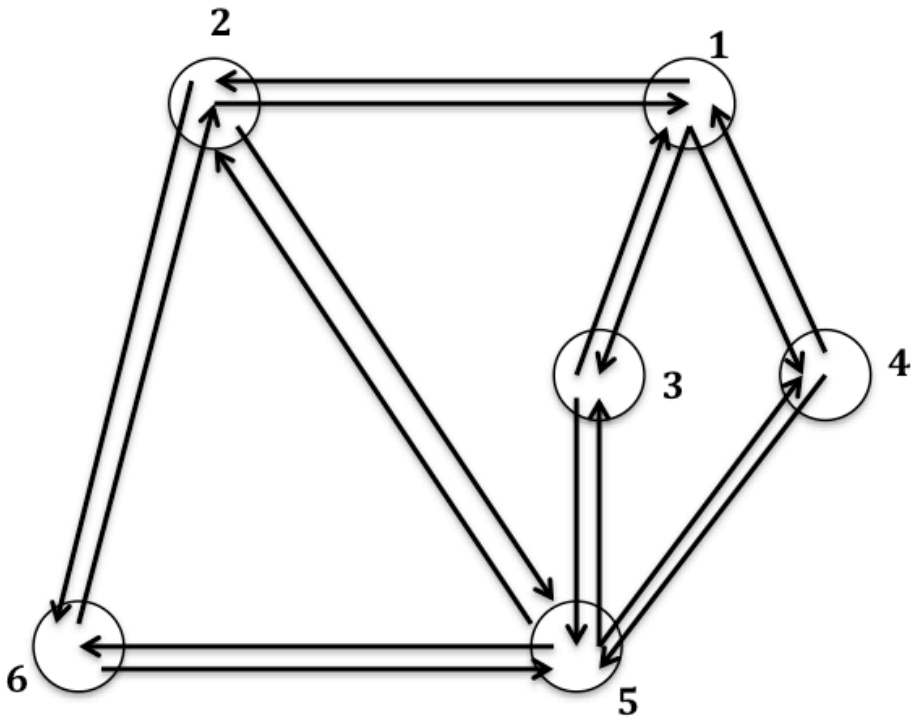

Fig. 1.6 The Directed Representation of a Bidirectional Graph

1.5 The DC Power Flow Model

The second topology analysis method proposed in this work utilizes the DC power flow model. The power flow problem consists of determining the steady state operating conditions of a known transmission network that is composed of transmission lines and transformers. The scheduled generation and the loads within the system are known and the power flow solution consists of obtaining the voltage magnitude and phase angle at each node of the system. The equations that represent the power flow at each bus in an alternating current (AC) system are nonlinear and given below in (1.1) and (1.2) [15],

$$P_i = \sum_{k=1}^{n} |E_i||E_k|\left[G_{ik}\cos\left(\theta_i - \theta_k\right) + B_{ik}\sin\left(\theta_i - \theta_k\right)\right] \tag{1.1}$$

$$Q_i = \sum_{k=1}^{n} |E_i||E_k|\left[G_{ik}\sin\left(\theta_i - \theta_k\right) + B_{ik}\cos\left(\theta_i - \theta_k\right)\right], \tag{1.2}$$

where,

P is real power

Q is reactive power

E is the bus voltage magnitude at node i or k

$G_{ik} + jB_{ik} = Y_{ik}$ is the ik^{th} term in the Y system matrix.

The Y matrix, or bus admittance matrix, is a matrix of admittances that represent a transmission network. An example admittance matrix can be seen in Table 1.2.

Table 1.2 Example Admittance Matrix

$Y_{1g} + Y_{12}$	$-Y_{12}$	0	0
$-Y_{12}$	$Y_{2g} + Y_{12} + Y_{23}$	$-Y_{23}$	0
0	$-Y_{23}$	$Y_{3g} + Y_{34} + Y_{23}$	$-Y_{34}$
0	0	$-Y_{34}$	$Y_{4g} + Y_{34}$

A connection of R +jX from bus 1 to bus 2 is equal to $1/(R + jX) = G_{12} + jB_{12} = Y_{12}$. Also, the subscript g represents a connection to ground or the reference bus [15].

The nonlinearity of the power flow equations requires that they be solved using a nonlinear method, such as the Newton-Raphson method. Nonlinear solvers are iterative processes. One method of obtaining solutions faster is by making approximations in the equations such that they become linear and can be solved directly. This process results in the DC power flow model. In the DC power flow, the equation for reactive power flow (1.2) is completely ignored. In (1.1), it is assumed that the voltage magnitudes in the system are exactly 1 pu at all times. Also, the resistance in the transmission network is neglected which results in all terms with G to disappear. Finally, it is assumed that the angle difference across any line or transformer in the system will be very small. This allows $\sin(\theta_i - \theta_k)$ to be expressed as $(\theta_i - \theta_k)$ in radians. The power flow on each line in the system using this method is shown in (1.3),

$$P_{ik} = \frac{1}{x_{ik}}(\theta_i - \theta_k) \quad . \tag{1.3}$$

This equation written in matrix form is given by,

$$P = Y\theta \tag{1.4}$$

where P is a vector of power injections, Y is the admittance matrix, and θ is a vector of bus voltage phase angles [15]. Equation (1.4) is solved in two steps. The admittance matrix is factored into LU components, where L is a lower left triangular matrix and U is an upper right triangular matrix where $LU = Y$. Forward and backward substitution is employed to solve for bus angles for a given set of power injections. For a dense system, LU decomposition takes $O(n^3)$ operations to complete and forward and backward substitution

takes $O(n^2)$ operations [16]. For a sparse system, the number of computations needed for LU decomposition is,

$$n + \sum_{i=1}^{n-1} r_i + \sum \left(r_i^2 + r_i \right) / 2 \quad , \tag{1.5}$$

where n is the number of buses in the system and r_i is the number of nonzero elements to the right of the diagonal in row i of the U matrix. For a sparse system, the number of computations needed for forward and backward substitution is shown in (1.6) which is $O(n)$ for a typical power system topology [17],

$$2n + 2\sum_{i=1}^{n-1} r_i \quad . \tag{1.6}$$

Every time a solution is desired, forward and backward substitution is required. However, factorizing the admittance matrix only needs to be done when the system topology changes. If the system network is altered by the status change of one line, four elements in the admittance matrix need to be modified. For example, if line bus 1 to bus 2 no longer exists in Table 1.2, elements $Y(1,1)$, $Y(1,2)$, $Y(2,1)$, and $Y(2,2)$ must be modified. The DC power flow allows rapid approximate solutions of real power flow in the system [15]. The DC power flow is utilized in this work to quickly analyze a power system topology.

1.6 CART Algorithm and Database

The program CART (classification and regression trees) produced by Salford Systems is a data-mining tool that can be used to analyze problems that contain a large number of variables. CART uses a procedure called binary recursive partitioning to build a decision tree (DT). Starting at the root node, simple questions called critical splitting

rules (CSR) are asked regarding one of the input parameters. The parameter selected for the critical splitting rule is called a critical attribute (CA). Each answer to the critical splitting rule question creates two branching nodes. Each branching node will have its own CSR regarding one of the input parameters. When any of the stopping criteria are met at a node, the node will not branch off. Nodes that do not branch off to other nodes are called terminal nodes that end the growth of the tree. Once all terminal nodes are reached the DT is complete and can be used to categorize new inputs. Given a matched set of input and output data, CART will determine its inherent input-output relationship in the form of a DT. This process is called DT training. Once training is complete, new input data can be dropped down the DT to generate the previously unknown output. Using this method, the historical PMU measurements and simulation results will serve as the necessary information needed to train the DT [18].

In order for CART to run an analysis it must first have a database. The database is usually composed of a learning dataset and a test dataset. The learning dataset is used to train the DT and the test dataset is used to test the DT. An example CART database can be seen in Table 1.3. A CART dataset contains a single class along with several features. The features act as input variables and the class is the categorical outcome. The CART database can have a maximum of 32768 features. Adding an additional feature to the example CART database would increase the database by 1 column. Additional rows may be added to the database to increase the amount of data samples included in the analysis. Each CART feature can be either continuous or categorical.

CART builds a DT by considering all possible univariate CSRs and selecting the CSR that makes the child nodes the purest. A node is considered pure if all the cases that

map to that node have the same class. CART then examines all the CSRs for the child nodes and continues making addition splits until a stopping criterion is met. CART uses the Gini criterion to determine the CSR at each node. The Gini impurity is a measure of the partition purity of the data. The Gini impurity at a node *t* is defined,

$$i(t) = \sum_{i,j} C(i \mid j) p(i \mid t) p(j \mid t), \tag{1.6}$$

where $C(i \mid j)$ is the cost of misclassifying a class j case as a class i case and $p(j \mid t)$ is the probability of a case being class j given that it falls into node t. The Gini splitting criterion is the decrease of impurity defined in (1.7) for a node t and a split s,

$$\Delta i(s,t) = i(t) - p_l i(t_L) - p_R i(t_R), \tag{1.7}$$

where p_L and p_R are the probabilities of sending a case to the left child node t_L and to the right child node t_R of node t respectively. The value of p_L can be computed as

$$p_L = \frac{p(t_L)}{p(t)}, \tag{1.8}$$

where $p(t)$ is the probability of a case existing in node t. CART will consider all possible splits and compute their corresponding purity improvement $\Delta i(s,t)$. The CSR with the largest improvement will be chosen as the CSR for node t. The process of growing the tree will continue until a stopping criterion is met. When a stopping criterion is met at a node the node will not branch off. Examples of stopping criteria include: when a node becomes pure, when all cases in a node have identical feature values, the DT has reached the maximum tree depth specified by the user, the size of a node is less than the minimum node size set by the user, or the best split s for node t has a purity improvement $\Delta i(s,t)$ smaller than the minimum specified improvement [19].

Consider the example database in Table 1.3. This example database holds a very simple input-output relationship that can be observed by examining the data. If input 1 is larger then 6.7 then the output is 1 regardless of the value of input 2. However, if input 1 is smaller then 6.8, then the output is dependent on the sign of input 2. A CART DT using the example CART database can be seen in Fig. 1.7.

The CART DT in Fig. 1.7 also shows a very simple input-output relationship. Here, if input 1 is larger then 6.75 then the output is 1 regardless of the value of input 2. However, if input 1 is smaller then 6.75, then the output is dependent on the sign of input 2. The example CART database shown in Table 1.3. The corresponding CART DT shown in Fig. 1.7 is an example of how CART can be used to generate a prediction model. In a real application the CART database could have many rows and the relationship within the data could be very complicated to determine. The DT that is generated by CART will always match the training data very well. In order to correctly judge the prediction accuracy a DT, some of the available samples must be withheld from the learning dataset to create a test dataset.

1.7 Literature Review

1.7.1 Network Flows

One of the primary contributions of this work is the proposed algorithm used to determine minimal cut sets within a power system to monitor system topology. The first network flow algorithm, developed by Ford and Fulkerson, finds a solution using augmenting paths [20]. The Edmonds and Karp algorithm extended this concept by

Table 1.3 Example CART Database

Output	Input 1	Input 2
1	1	0.01
0	1	-0.01
0	1	-0.3
1	1	0.4
1	1	0.3
1	1	0.2
0	1.5	-1
1	2	2
1	2.5	1.5
0	3	-0.73
0	3.5	-0.4
0	4	-3
1	4.5	0.6
1	5	0.8
1	5.5	4
0	6	-2.2
0	6.6	-2.3
0	6.7	0.2
1	6.8	-0.4
1	6.9	0.869
1	7	-0.3
1	7.1	-0.33
1	7.2	0.01
1	7.3	-0.99
1	7.4	-9.97
1	7.5	18
1	7.6	-3
1	7.7	0.12
1	8	-1

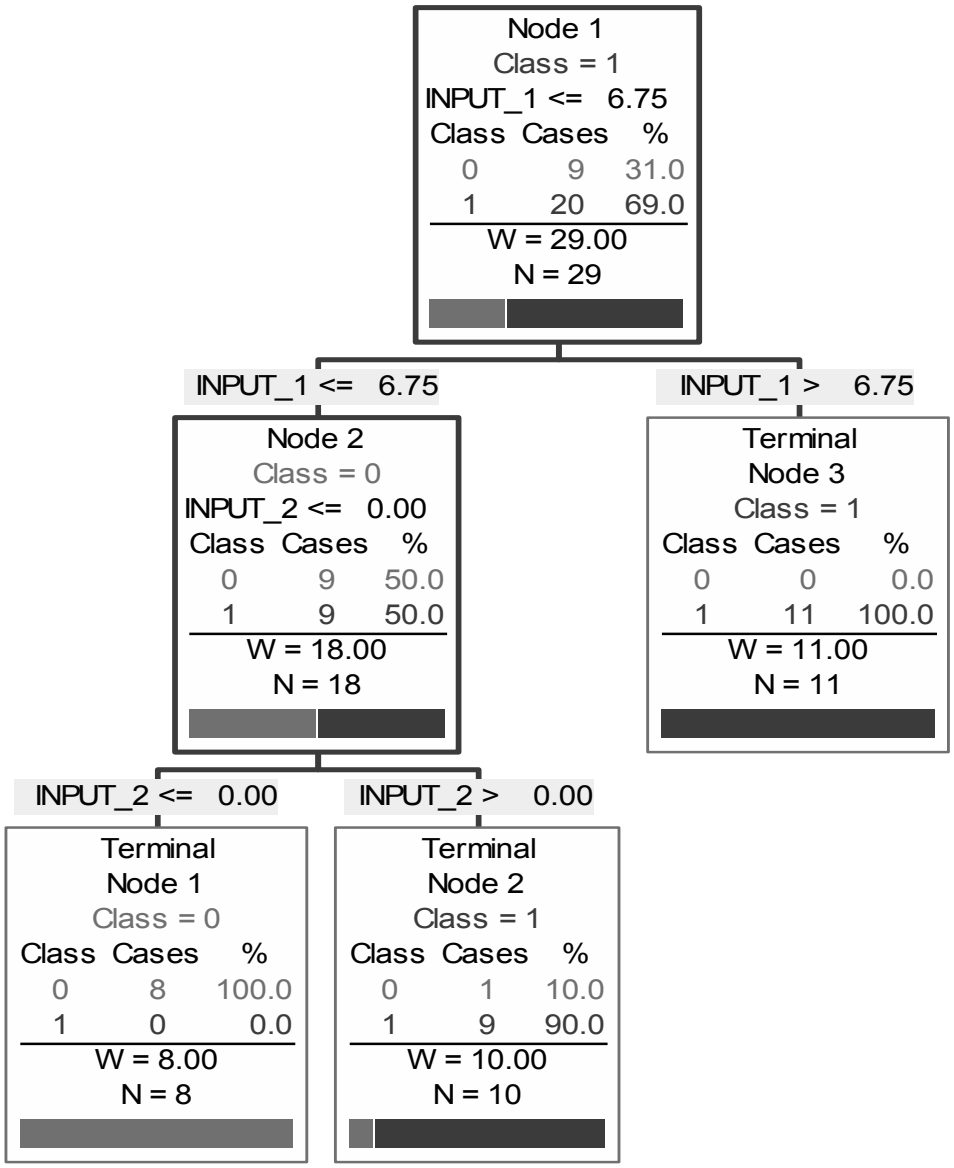

Fig. 1.7 Example CART Decision Tree

only using the shortest possible augmenting path each iteration to achieve a run time of $O(E^2V)$ [4]. Dinic proposed the used of search trees to find all the shortest paths of a specific length in one iteration [21]. Algorithms like the Karzanov algorithm are ideal for dense graphs while algorithms like the Goldberg and Tarjan algorithm are ideal for sparse graphs [4]. Network flows are utilized in image processing [22]. Reference [22] proposes a network flow algorithm similar to [21] which uses search trees to find paths of a shortest length. However, [22] proposes reusing the search trees in the next iteration. This algorithm has a worst-case complexity that is worse than similar algorithms but was found to have superior run time in practical applications [22]. This dissertation proposes a new algorithm for finding the maximum flow in a graph in the case were a graph topology has changed by the status of only a single edge and a new solution is desired. This work proposes a way of obtaining the new solution using the old network flow information to provide a significant computational advantage.

This work proposes the use of network flows to find minimal cut sets within a power system to provide enhanced situational awareness during major system disturbances. Other work that utilizes cut sets deal with cut set enumeration to determine communication system reliability. In [23, 24], minimal cut sets within a network are determined to define the reliability of a communication between two points in the network. Also, minimal cut sets have been used to identify the reliability of distribution systems [25]. A graph-algebraic approach has been proposed that uses a spectrum plot of the system Laplacian matrix to analyze system topology [26]. An eye-inspection metric is then applied to the spectrum plot to help interpret the results. This spectrum plot method is computationally intensive and may not be well suited for large system applications. This

dissertation presents an efficient method to monitor the topology of a power system, using graph minimal cut sets, which produces results with a very clear and concrete significance.

1.7.2 Critical Line Identification

Another major objective of this work is to determine critical lines within a given topology. In this work, a critical line is defined as a line that will disconnect two areas of the system if it were taken offline. In graph theory this is referred to as bridge detection [2]. A bridge in a graph is an edge that will disconnect two sets of vertices if removed from the graph. There are two different types of approaches, graph theory based and matrix based, that have been proposed that can address this problem [27, 28, 29]. Both [27] and [28] propose to be used to find bridges specifically within a power system network. One graph theory method involves the removal of cycles from a graph [27]. Recall that a path in a graph is a set of vertices and edges were no vertex or edge is repeated. A cycle is a path that begins and ends at the same vertex (creating a loop). In a cycle, there exist two paths between every pair of vertices. Therefore, a cycle cannot contain any bridges. Reference [27] presents an algorithm that finds and removes cycles from a graph. Once the process terminates the only remaining edges are bridges within the original graph. Both graph theory methods of [27] and [29] require $O(E)$ operations to complete. Matrix methods involve factoring some form of the network adjacency matrix to identify islands or bridges. Reference [28] proposes the triangular factorization of the bus-branch incidence matrix of a network to identify bridges. This method also requires $O(E)$ operations to complete. This dissertation proposes an alternative method of bridge finding in power

systems that utilizes standard operations used to solve a DC power flow study. The proposed method also achieves $O(E)$ complexity which is consistent with other algorithms.

Other related work considers minimum cut set determination using generalized line outage distribution factors [30]. When given a system topology and a set of lines, this method can determine if the set of lines chosen contains a cut set and which lines of the set are a part of that cut set.

Line outage distribution factors (LODF) deal with changes in line flows following the loss of a transmission line or transformer using the DC power flow model and are defined as,

$$d_{lk} = \frac{\Delta f_l}{f_k^0} \quad , \tag{1.9}$$

where d_{lk} is the line outage distribution factor looking at line l after an outage of line k, Δf_l is the change in MW flow on line l, and f_k^0 is the original flow on line k. If the current flow on lines l and k are known, the flow on line l after the loss of line k can be determined by,

$$f_l = f_l^0 + d_{lk} f_k^0, \tag{1.10}$$

where f_l^0 and f_k^0 are the initial flows on lines l and k, and f_l is the flow on line l with line k removed [15]. A power transfer distribution factor (PTDF) deals with the change in flow of a selected line caused by power injected at one point in the system and the same power withdrawn from a different point in the system. For a transaction of t MWs between points i and j, the impact on line l is given by the PTDF P_l^{ij} where

$$\Delta f_l = P_l^{ij} t \quad . \tag{1.11}$$

For a set of chosen lines in the system a matrix H_a can be formed. Assume the lines selected are l_1 to l_a. Define the impact on line l for a transaction of t MWs between the endpoints of line l as P^l. Then matrix H_a takes the form

$$H_a = \begin{bmatrix} 1-P^{l1} & \cdots & -P^{la} \\ \vdots & \ddots & \vdots \\ -P^{la} & \cdots & 1-P^{la} \end{bmatrix} . \tag{1.12}$$

If the matrix H_a is singular then the set of chosen lines contains a cut set [30].

One very important observation discussed in [18] has to do with the effect of singleton cut sets (bridges) on the PTDFs in the network. Consider the line l in a power system that connects two subnetworks together (line l is a bridge). Also consider a transaction of t MWs between the terminal buses of line l. Then the line l is a minimal cut set, if and only if

$$P^l_{l_m} = \begin{cases} 1, & l_m = l \\ 0, & l_m \neq l \end{cases} . \tag{1.13}$$

This can be extended further. Consider two subnetworks N- and N+ that are connected by line l. For any transaction from point i to point j, where point i is in N- and j is in N+, the PTDF for this transaction on line l will be 1. The method used to detect critical lines in this work takes advantage of this property and is discussed further in Chapter 2.

1.7.3 PMU Line Outage Detection

The other portion of this work deals with transmission line outage determination. This problem is addressed in [6, 31-37]. The problem can be broken down into *detecting* when an outage occurs and then *matching* the outage to a specific line within the system.

One of the earliest proposed methods to address this problem is discussed in [6]. A method commonly used in edge detection is used to detect when an event has occurred. This method uses a sliding window over the low-pass filtered phase angle measurements. The angle at measurement n is compared with the measurement n-i. The variable i determines the width of the window. A value of $i = 40$ measurements was used in [6]. Once the value of n-i exceeds the threshold τ, the method determines that an event has occurred. A value of $\tau = 0.57^{o}$ was used in [6]. The window continues to move through the data until the value of n-i begins to decrease. This signals that the event has ended and *matching* can begin. The maximum value of n-i within the event interval determines the change in quasi-steady state voltage phase angle caused by the change in topology. The change in voltage phase angles measured by PMUs within the system creases a vector of phase angle changes. This vector will have one dimension for every PMU in the system. An approximate DC power flow model of the system is used to determine the change in voltage phase angles that would occur for the loss of any transmission line within the system. The vector of angles recorded from PMUs is compared to all vectors found through simulation. Finally, the vector from simulation that matches the vector from PMU measurements the closest is declared the line that was lost.

Later works expanded on the issue of *matching* [31-37]. Reference [31] applies the idea proposed in [6] to identify a simultaneous double line outage. This method still requires an exhaustive search to identify the lines lost but only requires a small number of PMUs to be available. Reference [32] uses a support vector machine approach to match outaged lines. Another work extends previous methods to work when observing only a particular area of the system and also deals with the cases of islanding caused by the line

outage [33]. The work in [34] considers the case of a fully observable internal system connected to an external system that contains some PMUs. The method uses measurements from PMUs as well as power flow changes along tie lines in an integer programing problem to determine lines lost within the external system. Also, several studies consider the case where the system is fully observable by PMUs [35-37]. These methods include applying alternating direction method of multiplies [35], a global stochastic optimization technique based on cross-entropy optimization [36], and Gaussian Markov random fields [37] to match outaged lines using PMU measurements.

Of all the work done in this area only [6] discusses the problem of outage *detection*. Reference [6] uses real PMU data for testing while the remaining studies only test using a simulated system. In a simulated system the problem of detection is not present. During a simulated line outage, prior to the line outage, all the system parameters have no variation, which makes the beginning of the event obvious. Recall the work in [6] addresses the problem of detection by using a small threshold (τ) to detect the beginning of an event. However, this method could not be adapted to the Gustav event in the Entergy power system. The issue arises from the PMU data itself. The work done in [6] shows the PMU data during an outage of a 500 kV line carrying approximately 1072 MW in the Tennessee Valley Authority (TVA) system and can be seen in Fig. 1.8. In contrast, the PMU data during an outage of a 500 kV line during the Gustav event is shown in Fig. 1.9.

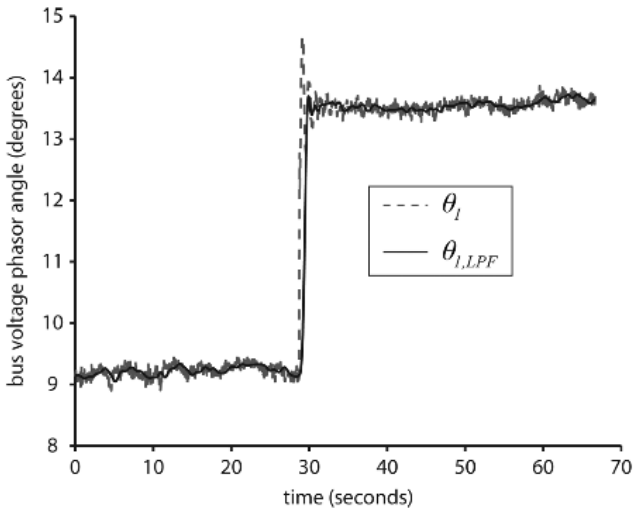

Fig. 1.8 Filtered and Unfiltered PMU Data During a 500 kV Line Outage [6]

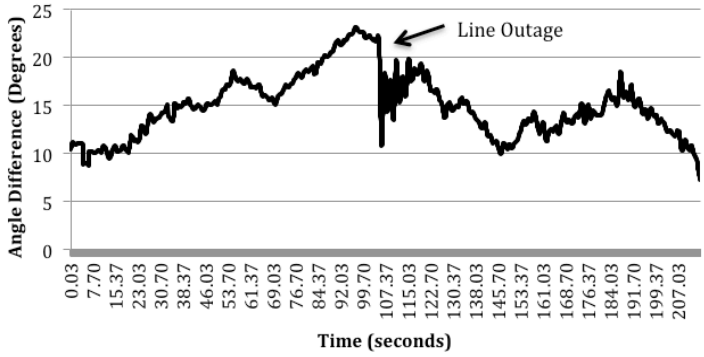

Fig. 1.9 Voltage Phase Angle PMU Data Showing a 500 kV Line Outage During the Gustav Event

It is apparent by comparing the different measurements that the phase angle in the Entergy system is far more erratic than the measurements in the TVA system. The study in [19] arbitrarily chose a small value of τ and their method works well with their data. However, the measurements taken during the Gustav event in the Entergy system are significantly different. In order to find an acceptable event detection method using the En-

tergy data, a different approach is needed. This work proposes the use of a trained DT approach to provide a robust method of line outage detection using PMU measurements.

1.7.4 Applications of Decision Trees

Applying the DT data-mining techniques, available through the use of CART, to analyze complex problems is not a new idea. DTs have been applied to many different fields of study. One medical study conducted in Taiwan, used CART analysis to predict the survival of patients that were diagnosed with liver cancer [38]. A predictive model was desired to be able to provide information to patients for understanding treatment outcomes. The study used a total of nine features (inputs) in the study. These variables were age, tumor size, tracking period, gender, clinical stage, undergoing surgery, radiotherapy, chemotherapy, and transcatheter arterial embolization. The class (output) was whether the patient had survived after 5 years. The study used records of 136 patients with liver cancer as training data for the decision tree training. The study then used the records of 91 patients with liver cancer as the database to test the accuracy of the tree. The CART model was able to correctly predict the 5-year survival of the 91 patients 74% of the time [38].

Sometimes the selection of features to use in DT training is not clear. The work done in [39] uses a DT and simulated PMU measurements to detect an island formation around a distributed generator. The study was interested in the transient signal measured by PMUs during island formation. Data from PMUs are a series of measurements over time. Simply providing CART with a series of measurements can make it very difficult to train a DT with acceptable performance. Instead, intervals within the series are selected

around periods of interest. A discrete wavelet transform was used to determine the wavelet coefficients for each interval. Only these coefficients were then used as features in the DT datasets. This study showed a trained DT could identify an island formation around a distributed generator with a classification accuracy of 98%. Similarly, the work done in [40] uses a DT and simulated PMU measurements to detect high impedance faults. In this study, the harmonic content within each interval was extracted using a fast Fourier transform. The harmonic content was used as features in the DT datasets. This study showed a trained DT could identify a high impedance fault with no misclassifications for 100 simulated test cases. The work done in this dissertation uses a similar strategy for feature selection.

1.8 Objectives

This work is intended to provide an efficient method of monitoring the topology of a power system to provide enhanced situation awareness during significant topology related system disturbances. The approach will use an efficient and effective topology analysis method to identify the minimum number of lines connecting important points in the system as well as critical lines (bridges). The method will also use a DT approach to identify transmission line outages to allow accurate topology information to be maintained in the event of SCADA system telemetry deficiencies. The primary sub-tasks of this research, with objectives and a description of the approach adopted for each sub-task, are given below:

1. Present and explain the strategy of applying network flow solutions to a power system. The network flow algorithm used to update the minimum cut sets after the

loss of a transmission line will be presented. A detailed example of the proposed method will be presented with a recreation of the Gustav test case. Also, the value of the proposed approach will be examined by discussing two other recent major system disturbances.

2. Present and explain the OLR algorithm used to detect critical lines. The OLR algorithm will then be verified using two test systems. The first system is a small system of 16 buses. This small system will allow the reader to easily validate that each case run by the OLR algorithm is correct. The next system is a larger 118 bus IEEE test system. Finally, the Gustav event will be recreated in the Entergy system model and the OLR algorithm will be run after each tie line trip to illustrate the significance of having the OLR output during a real event.
3. Create a trained DT for use in line outage detection or SCADA outage verification. First, the features to be used in the DT datasets will be selected and explained. The learning and testing datasets will be created using part simulation data and part real PMU data. All line outages in the datasets will come from simulations while all the non-event data will come from real PMU measurements. The DT will be trained with the learning dataset and then tested using the test dataset.
4. Test the accuracy of the trained DT using real PMU measurements. The PMUs in the Entergy system captured the loss of a 500 kV transmission line that can be seen in Fig. 1.9. This trained DT will be applied to this event to show the reliability and accuracy of the approach.

1.9 Organization of This Report

Following the introductory material in the first chapter, this report presents the explanation and validation of the topology algorithms in Chapter 2, as well as a discussion of their breadth of application. Then, details of training and validating the DT used to detect line outages are presented in Chapter 3. Next, conclusions are drawn in Chapter 4. Also, the Matlab code implementation of the network flow algorithms can be seen in Appendix A, and the Matlab code implementation of the OLR algorithm can be seen in Appendix B.

2. TOPOLOGY ANALYSIS

2.1 Network Flow Analysis

This work proposes the use of network flow algorithms to determine the minimum number of lines connecting important locations within a power system. The important locations used by the algorithm are buses defined by the user. Recall that a network flow algorithm can determine the maximum flow between a source and a sink within a directed graph. If the edges within the graph have capacities of exactly one unit of flow, then the maximum flow found within the graph will be equal to the minimum number of edges that must be removed in order to guarantee disconnecting the source from the sink [2]. This work proposes the selection of a single reference that will act as the source in all network flow cases. A number of sinks throughout the system are selected by the user to create a reasonable number of network flow cases. The solution of each case represents the minimum number of transmission lines or transformers that must be lost in order to disconnect that specific sink from the reference. A realistic and detailed example of this description is presented in Section 2.1.2. This work is included in the future publication [41].

The locations of sources and sinks within the network must be defined before a network flow algorithm can be applied to the system. A large power system may have 10,000 buses. If a single bus were selected as the source, and every other bus serves one time as a sink, this would require 9,999 different cases to be run using the network flow algorithm. Not only is this time consuming it is also unnecessary. Instead, only a small subset of buses needs to be selected. For example, a single centrally located bus with high

degree can be selected as the source. These buses tend to be important to the system in some way such as large generators, critical loads, large substations, or PMU sites. The sinks in the system can be assigned to other important buses throughout the network. Such as setup would guarantee that the network flow solutions would indicate a problem if outages in the system begin to isolate any important buses from the source. Using this method, it is possible that an island could form that does not include any of the chosen sinks, which would be invisible to this approach. However, the intent of this approach is to observe major system separations and the absence of a sink within the island implies such an event is no a significant threat to the operation of the system.

2.1.1 Algorithm to Update a Network Flow Solution

Consider a single case using the selected source and one sink. The first network flow solution to determine the maximum flow must be done using one of the network flow algorithms presented previously. Generating a new network flow solution is only required when the topology of the system changes in some way. When the topology of a power system changes, it is usually due to the outage of a single transmission line. This work presents a method of modifying an existing network flow solution after the loss of a single branch, to reduce the computational burden of the algorithm and to allow more sink locations to be selected. Consider the bidirectional graph in Fig. 2.1.

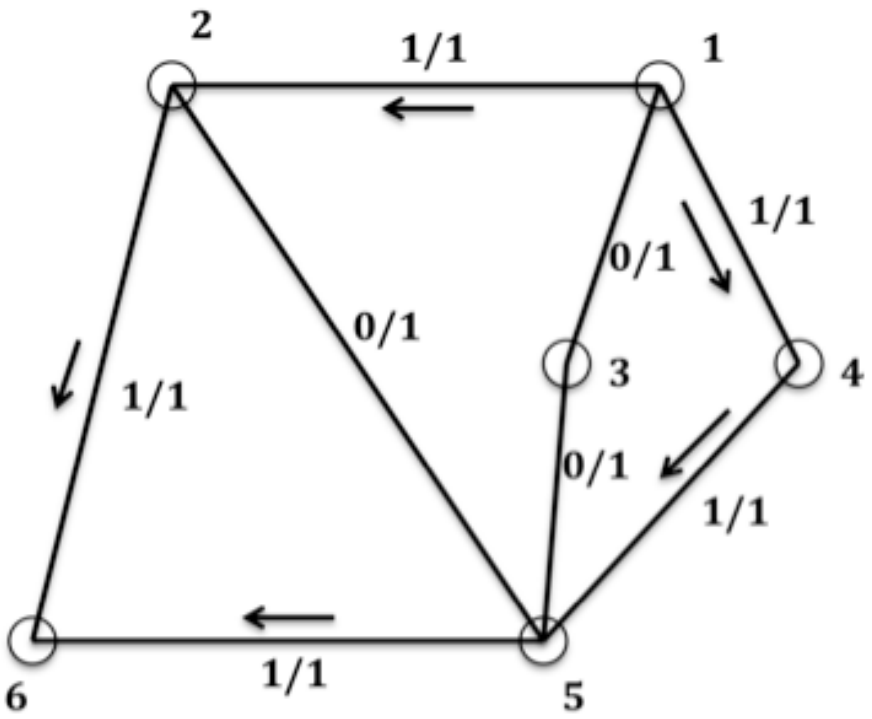

Fig. 2.1 The Maximal Flow in a Network with All Edge Capacities of One

The Edmonds and Karp algorithm was applied to this network with a source at vertex 1 and a sink at vertex 6. It is clear that the minimum cut set connecting vertex 1 and vertex 6 is of size 2, which is equal to the maximum flow in this graph. Recall the criteria a network flow solution must satisfy in order to be correct (1, 2, and 3).

1. For every edge, the flow across an edge cannot exceed the capacity of that edge.
2. For every vertex other than the source and the sink, the sum of the flows entering a vertex must equal the flow exiting that vertex.
3. The flow from source to sink must be maximal. The flow is maximal, if and only if, the graph contains no more augmenting paths between the source and the sink.

When an edge is removed from this graph there are three different possible cases that could occur. In all cases, a new solution is reached when the capacity and flow on the removed edge becomes zero while criteria 1, 2, and 3 are still satisfied.

- Type Zero – *The edge being removed from the graph has zero flow in the current network flow solution.* This is the trivial case. If the edge being removed from the graph has no flow then the capacity of that edge can be set to zero and the new solution is reached. Because no flow was added to the graph, criteria 1, 2, and 3 must still be satisfied and the solution is correct. An example of a type zero edge is edge 1 to 3 in Fig. 2.1.
- Type One – *The edge being removed from the graph has flow in the present network flow solution and is part of a cut set of minimum size*. The removal of such an edge will reduce the maximum flow in the graph by one. An example of a type one edge is the edge 2 to 6 in Fig. 2.1. In the old network flow solution the flow is maximal. There are no more augmenting paths between the source and the sink and every cut set of minimum size is completely saturated. Also, there will be no augmenting paths between the endpoints of the removed edge in the same direction as the edge flow. For example, all paths from vertex 2 to vertex 6 must cross at least one edge that is part of a minimum cut set. However, all minimum cut sets are saturated. Therefore, if no augmenting path can be found between vertex 2 and vertex 6, then the edge belongs to a cut set of minimum size. Then, the solution can be updated by finding a single valid path that starts at the sink and ends at the source that includes the removed edge. A flow of one is added along this path and effectively reduces the maximum flow in the network by one. The addition of

this flow will also eliminate the flow on the removed edge and the capacity of the removed edge can be set to zero. Criteria 1 and 2 remain satisfied because the only flow added to the system was along a valid path that spanned from the source to the sink. The criterion 3 is satisfied when the maximum flow in the graph is reduced by one. An updated solution for the removal of edge 2 to 6 can be seen in Fig 2.2. The direction and location of the flow added to the system is shown. The path used to adjust the solution is found using the BFS path finding algorithm [14]. The BFS algorithm runs in $O(E)$ time. The total time to update a solution for the loss of a type one edge is also $O(E)$.

- 3. Type Two – *The edge being removed from the graph has flow in the current network flow solution but is not part of a cut set of minimum size*. An example of a type two edge is the edge 1 to 2 in Fig. 2.1. The removal of such an edge must not reduce the maximum flow in the system. Because this edge is not part of a minimum cut set there must exist a path from 1 to 2 that does not require crossing a saturated edge. This path can be used to create a cycle of flow that will eliminate the flow across the removed edge while still maintaining the maximum flow within the graph. An updated solution for the removal of edge 1 to 2 can be seen in Fig. 2.3. Criterion 1 is satisfied because the added flow does not need to violate any edge capacities. Criterion 2 is satisfied because adding flow around a closed loop does not violate the conservation of flow. Finally, criterion 3 is satisfied because the maximum flow in the graph did not change. The path used to adjust the solution is found using the BFS path finding algorithm. The total time to update a solution for the loss of a type two edge is also $O(E)$.

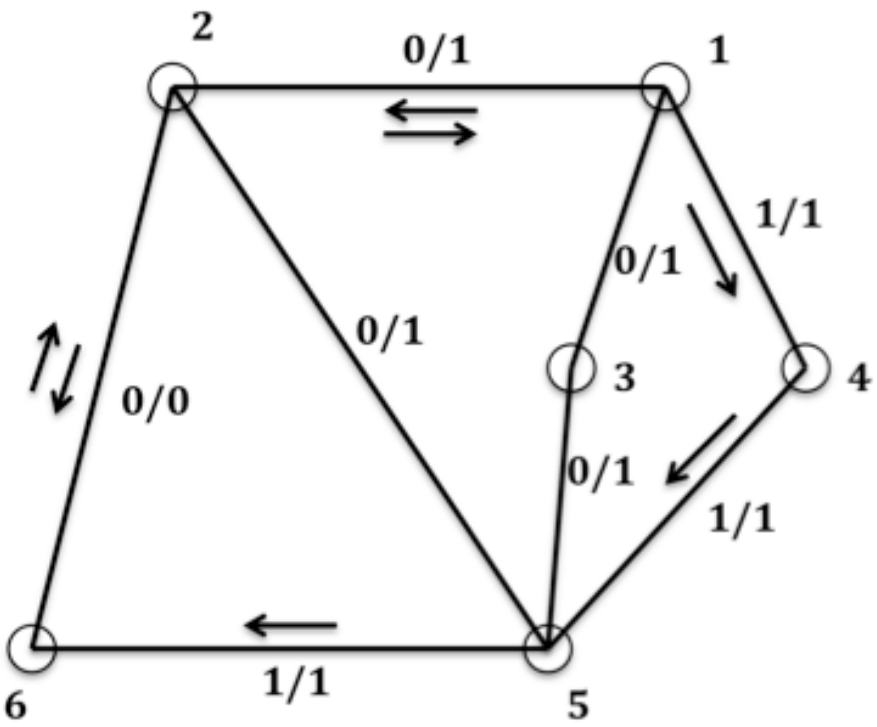

Fig. 2.2 The Updated Solution for the Loss of the Type One Edge 2 to 6

Two steps are needed to determine the type of each edge within a network flow solution. If there is no flow on the edge it must be type zero. If the edge in question contains flow, the edge type can be determined using a single path search. If an augmenting path can be found between the endpoints of the edge in question, in the same direction as the edge flow, then the edge must be type two. If no augmenting path can be found, the edge must be type one. Once the edge type is determined the new solution can be updated after a single path search. This algorithm requires $O(E)$ operations to update a solution which is computational less expensive than creating a new solution using any known algorithm.

Once the reference and sink locations are selected, the initial network flow solution for each reference-sink pair is found using Edmonds and Karp. Following a trans-

mission line outage during system operation, the solution for each reference-sink pair is updated for the lost edge using the previous network flow solutions. The new values of minimum cut sets can then be displayed over a visual representation of the system to provide the operator, at a glance, the approximate area that was stressed by the most recent line outage. The values of minimum cut set provide an approximate indication as to the severity of the stress when compared to their value during normal system operation. The sinks that have reduced minimum cut sets following an outage provide an approximate affected area. As states previously, only one sink needs to be in an affected area to indicate system stress. If one sink lies within an affected area, a more specific group of buses can be found by finding the disjoin bus sets associated with that specific reference-sink cut set. For example, consider the network in Fig. 1.5. The source is at node 1 and the sink is at node 6. The cut set separating the source and sink in this example are lines 1 to 2, 3 to 5, and 4 to 5. This cut set partitions the system into two disjoint node sets (1, 3, 4 and 2, 5, 6). This information can more accurately indicate the size of the area being pulled away from the grid. A cut set and corresponding disjoin sets can be found using a BFS of the network flow solution. Such an algorithm is described in [2, pg. 70] and runs in $O(E)$ time.

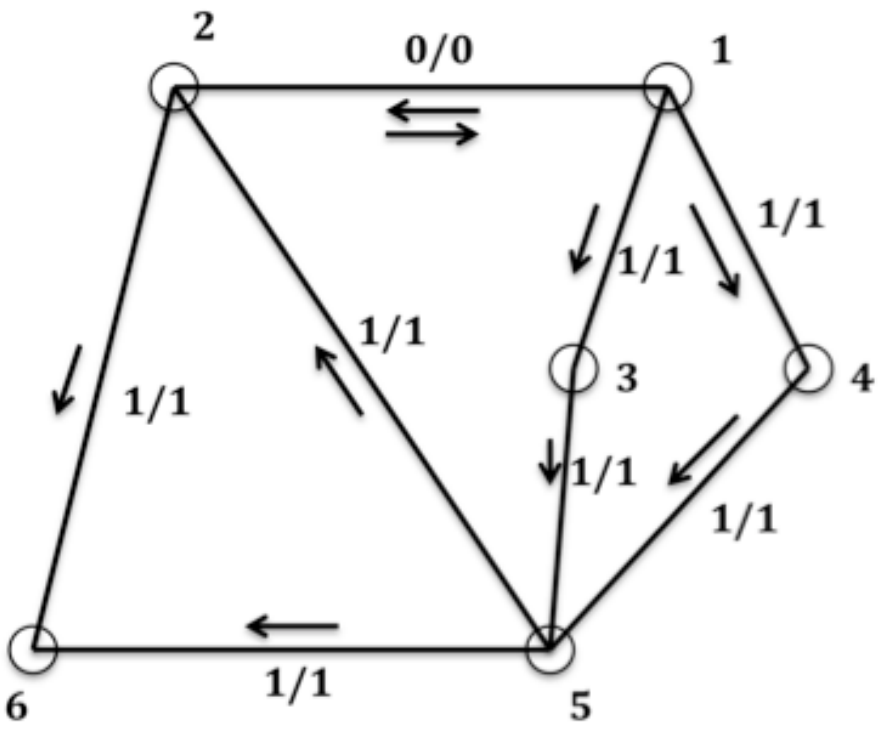

Fig. 2.3 The Updated Solution for the Loss of the Type Two Edge 1 to 2

2.1.2 Network Flows Applied to the Gustav Test Case

A network flow solution can provide important information to system operators during significant system disturbances. In 2008, hurricane Gustav made landfall off the coast of Louisiana. Over the course of several hours a large electrical island was formed around Baton Rouge and New Orleans following the loss of 14 transmission lines [12]. Network flows were applied to this system during a recreation of the island event to illustrate the information that would have been available to system operators following the loss of each transmission line. This recreation was conducted using a 20,000-bus operations planning model of the system. The network flow algorithms were implemented in Matlab. The Matlab implementation of the network flow algorithms can be seen in Ap-

pendix A. The source used in this study was located at the PMU site at El Dorado. El Dorado is a PMU site that is centrally located within the system. Ten sinks were used in this study. The sinks include Sterlington, Rodemacher, Coughlin, Scott, Cecelia, Fancy Point, Bogalusa, Willow Glen, Waterford, and Ninemile. The locations of Willow Glen, Waterford, and Ninemile lie within the islanding area. The results of the network flow algorithms following the loss of each transmission line can be seen in Table 2.1. The first column in Table 2.1 indicates the most recent line outage (in the same sequence as in Table 1.1) while the remaining columns indicate the minimum cut sets between any sink (node name given in the table) and the reference following that outage. It can be observed that the system outages only have effect on a limited number of locations. Also, after the 13th line outage the size of the minimum cut set between the reference bus and the three sinks within the island, Waterford, Willow Glen and Ninemile, reduces to 1, indicating that only one remaining path exists and the loss of this path will cause an island to form. In the subsequent row in Table 2.1, which shows the results for the loss of the 14th line, one observes that the size of the minimum cut set between the reference bus and the sinks Waterford, Willow Glen and Ninemile reduces to zero, confirming that an island has formed.

The size of the minimum cut sets within the system before the event began can be seen in Fig. 2.4. The minimum cut sets at 1:30 PM (after the 12^{th} outage) and at 2:10 PM (after the 13^{th} line outage) during the event can be seen in Fig. 2.5 and Fig 2.6 respectively. The numbers next to each node indicate the minimum cut set to that node as the sink.

Table 2.1 Network Flow Solutions Following the Loss of Each Transmission Line During the Gustav Event

Line Lost	Size of Minimum Cut Set with Node Shown as Sink									
	Sterlington	Rodemacher	Coughlin	Scott	Cecelia	Fancy Point	Bogalusa	Willow Glen	Waterford	Ninemile
Before Outages	3	4	3	6	3	3	3	6	6	4
1st	3	4	3	6	3	3	3	6	5	3
2nd	3	4	3	6	3	3	3	5	5	3
3rd	3	4	3	5	3	3	3	5	4	3
4th	3	4	3	5	3	3	3	4	4	3
5th	3	4	3	5	3	3	3	4	3	2
6th	3	4	3	5	3	3	3	4	3	2
7th	3	4	3	5	3	3	3	4	3	2
8th	3	4	3	5	3	3	3	4	2	2
9th	3	4	3	5	3	3	3	4	2	2
10th	3	4	3	5	3	3	3	3	2	2
11th	3	4	3	5	3	3	3	2	2	1
12th	3	4	3	4	3	3	3	2	2	1
13th	3	4	3	4	3	3	3	1	1	1
14th	3	4	3	4	3	3	3	0	0	0

The figures 2.4, 2.5, and 2.6 also show the approximate geographical locations of the selected nodes within the system. A comparison between Fig. 2.4 and 2.5 shows that after the 12th line outage the minimum cut sets to nodes within the island area have significantly reduced. This implies the cumulative effect of the outages in the system have been to stress the intertie between the southeast area and the rest of the system. In Fig. 2.6 the system has reached a critical state where only one line outage is needed in order to form and island around 3 of the selected sink locations. At 2:49 PM the 14th outage oc-

curs and the island forms. During the real event, system operators were unaware of the risk of island formation and did not determine an island existed in the system until 20 minutes after island formation [12]. However, the network flow solutions provide a strong indication of the effect that each outage has on the topology of the system.

The network flows were run on a desktop computer using a 2.93 GHz Intel Core i7 quad core processor. The program determines the initial network flow solution using the Edmonds and Karp algorithm. A single network flow solved using Edmonds and Karp requires approximately 0.6 s using the 20,000-bus system. Following the loss of each transmission line, each old solution is updated using the proposed method described previously. Updating a solution for the removal of a type one or type two edge requires approximately 0.06 s. Updating a solution for the removal of a type zero edge requires only approximately 0.002 s. Also, because power systems are extremely sparse, the number of edges that contain flow within a given case is very small. For example, in the Entergy system, only about 4% of edges contain flow in each case. Therefore, using the proposed approach provides a significant advantage in this application. This shows that network flow algorithms can be applied to very large power systems and still respond quickly following changes in system topology.

2.1.3 Advantages of Using Network Flow Monitoring

Network flow monitoring provides unique information about the system during a variety of situations. Some of the unique benefits of using the application proposed in this work are highlighted in this section.

Fig. 2.4 The Maximum Flow to Selected Sinks Within the Entergy System Prior to the Event.

Fig. 2.5 The Maximum Flows to Selected Sinks Within the Entergy System at Approximately 1:30 PM During Hurricane Gustav

Fig. 2.6 The Maximum Flows to Selected Sinks Within the Entergy System at Approximately 2:10 PM During Hurricane Gustav

During normal operations, power systems are monitored with a variety of real-time tools. For example, in the MISO (Midcontinent Independent Transmission System Operator) system, the supervisory control and data acquisition system (SCADA) system takes measurements throughout the system, such as breaker status and voltage magnitudes, and reports them back to the control room. State estimation is run every 50 seconds using the SCADA data. The state estimate solution generates a power flow case that describes the current operation of the system that is used for real-time contingency analysis. There are over 40,366 buses and about 8,300 contingencies in the MISO network model [42]. The MISO system runs two different implementations of contingency analysis, Quick CA (contingency analysis) and Full CA. Each implementation of CA analyzes a list of contingencies by solving the post contingency state using the fast decoupled power flow. The full list of contingencies is solved every 4 minutes using 8 processors in parallel. A smaller list of contingencies is run in the Quick CA that runs every 50 seconds [42].

The proposed graph theory based method in this work can provide assistance to system operators under a variety of scenarios were different amounts of information are available. During normal operation, the existence of problems in the system is often identified by contingency analysis. Power flow computations have a computational complexity of approximately $O(n^{1.2})$ using linear methods and $O(n^{1.4})$ using Newton's method for a single contingency, where n is the number of buses in the network [43]. A system with a large number of components requires many contingencies to be considered (such as the MISO system) which causes CA studies to take several minutes. In contrast, updating a single reference-sink case for the loss of a type one or type two edge takes $O(n)$ opera-

tions (E is approximately equal to n in large power systems). Recall that power systems are very sparse and most edges in a given solution contain no flow (less than 4%). Therefore, the computations needed to update all cases following a line outage is approximately $0.04sn$, where s is the number of sinks in the system and $s << n$. This allows the method to provide immediate indication of system degradation to operators following an outage. This information can be used in conjunction with live SCADA measurements to take immediate actions, to influence which contingencies are handled first in the next iteration of the real-time contingency analysis system, or help visualize the results of a CA study.

There are times during operation when state estimation, and therefore real-time contingency analysis, is unavailable either from convergence failures or software problems. For example, on August 14th, 2003, a major widespread blackout occurred effecting parts of the Northeast and Midwest United States and the Canadian province of Ontario [44]. Transmission line outages began at approximately 3:05 PM. By 3:46 PM, the system had lost a total of five 345 kV lines and fifteen 138 kV lines representing the loss of several important Ohio interties. At this point, analysis of the event indicated that the blackout still could have been avoided if around 2 GW of load was dropped in the Cleveland-Akron area. At 4:05 PM, another 345 kV line was lost that lead to the blackout about 5 minutes later. One of the major causes of this event was a software bug that affected the FirstEnergy Corporation (FE) control room in Ohio. The software bug effectively disabled the alarm system that was integrated into all system monitoring tools. “FE operators relied heavily on the alarm processor for situational awareness, since they did not have any other large-scale visualization tools such as a dynamic map board. The operators would have been only partially handicapped without the alarm processor, had they

known it had failed [44].” Also, just prior to the event, incorrect telemetry data disabled the state estimator operated by MISO. An operator corrected the problem but forgot to restart the monitoring tool to run every 5 minutes. As a result of these issues, system operators were unaware of major system problems during the event [44]. Although the alarm processing system failed, real-time information and measurements for the FE system were still being collected and available. The network flow analysis method presented in this work is only reliant on system topology information to remain operational. During similar situations, a minimum cut set visualization method could still function and enhance situational awareness in the absence of state estimation or provide indications of malfunctioning systems.

Lastly, it is common for neighboring systems to not be monitored with state estimation or contingency analysis. As a result, serious issues along the boundary between multiple systems can arise because no single operator is aware of all component failures. For example, on September 8th, 2011, a blackout affected parts of Arizona and Southern California in the southwest United States, as well as Baja California, Mexico [45]. This event was initiated by the loss of a single 500 kV line that transported power from Arizona, through the Imperial Irrigation District (IID), into the San Diego area. The outage occurred at approximately 3:27 PM. In less than a minute after the outage, two transformers in the IID system overloaded and tripped offline. Following these outages, over the course of about 10 minutes, 5 transformers and 4 transmission lines are lost which disconnect the San Diego area from the rest of the grid. Later investigations found “Affected transmission operators have limited real-time visibility outside their systems, typically monitoring only one external bus. As a result, they lack adequate situational awareness of

external contingencies that could impact their systems" [45]. For example, after the 500 kV line outage, the California Independent Transmission System Operator (CAISO) had partial visibility into IID's system, but could not see that two transformers had overloaded and tripped. Likewise, IID operators did not learn of the 500 kV line outage in real-time. During the event, different system operators observed large changes in flows into their systems, but were unable to understand the cause or significance of the changes [45]. One distinct advantage that the application of network flows has over traditional monitoring systems is the ability to easily use a large system model. Increasing the area analyzed by state estimation or contingency analysis is a large undertaking. The increase in area requires many more component parameters and models to be maintained as well as greatly affecting the solution time. However, network flows can be applied to a very large system, as shown in the previous section, while still providing results quickly, and can provide operators with basic topology information about areas not being monitored by any other means.

2.2 One Line Remaining Algorithm

The network flow analysis outlined in the previous section, tracks changes in minimal cut sets between user-defined system locations. The changes in a minimal cut set over time can be a strong indicator of a problem with the system. However, the only time it is certain the system cannot survive any line outages without a problem, is if a minimal cut set is found to be of size 1. To address this situation, this thesis proposes an alternative method, using only standard power systems matrix operations, of detecting when critical lines in the system are created. This algorithm is called the one line remaining

(OLR) algorithm. In this work, a critical line is defined as a line that will disconnect two areas of the system if it were taken offline. The OLR algorithm can identify critical lines in a power system following a transmission line or transformer outage, while also determining the area that would island following the loss of any critical line in the system. The OLR takes advantage of the properties of singleton cut sets discussed in Chapter 1. An example power system is shown in Fig. 2.6.

In Fig. 2.6, if line *h* is taken out of service, $P_l^{ij} = 1$. This is because all the power flowing from point *i* must pass through line *l* in order to reach point *j*. Point *i* can be placed at any point in area 1 and P_l^{ij} will not change. Therefore, if it is known what two areas a critical line link together, PTDFs can be used to determine which line is the critical line. Consider that new critical lines can only appear in the system when the topology changes.

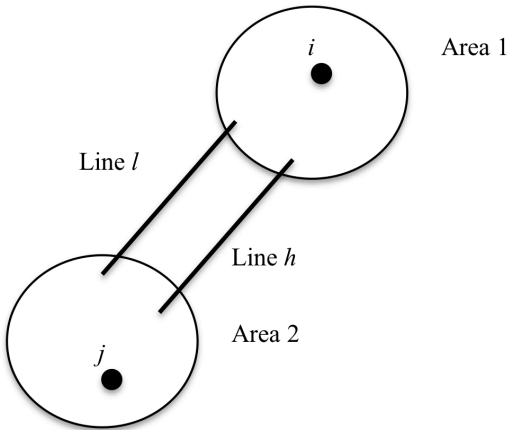

Fig. 2.6 Example Power System with Two Tie Lines

For a line outage, if new critical lines have appeared in the system then the endpoints of the outaged line must be in different areas. Consider the system in Fig. 2.6. Line *l* only becomes a critical line following the loss of line *h*. Also notice that the endpoints

of line *h* exist in different areas that would be disconnected if line *l* was lost. The OLR algorithm uses this knowledge to find the new critical lines in the system following each line outage.

For each line outage, the OLR uses a DC power flow study to compute the system line flows for power injections at the endpoints of the outaged line. If any lines in the system carry all of the power injected, those lines are identified as critical. For each critical line, the buses that are in the area of the system that is now vulnerable to islanding can be determined by using the voltage phase angles from the DC power flow study.

2.2.1 Algorithm Explanation

The OLR must be initialized before it is applied. Initialization is done by defining the system structure in the form of the system admittance matrix. When constructing the system admittance matrix all generators, loads, and shunt elements are ignored since the purpose of this algorithm is to examine the topology of the network. All branches in the network are represented with a reactance of 1 pu. By doing this, the admittance matrix represents a uniform weighted graph of the system topology. Consider the 4 bus system shown in Fig 2.7. The initialization of the 4 bus system in Fig. 2.7 can be seen in Fig. 2.8.

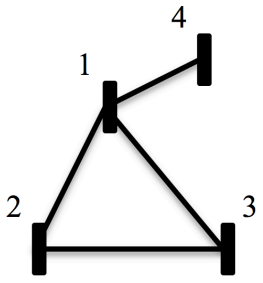

Fig. 2.7 Simple 4 Bus Power Network

$$-j\begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 2 & -1 & 0 \\ -1 & -1 & 2 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix}$$

Fig. 2.8 Admittance Matrix of the Network Shown in Fig 2.7

This step only needs to be done the first time the OLR algorithm is initiated. When a branch in the system is lost the OLR algorithm is invoked. The admittance matrix must then be updated to reflect the lost branch. Assume the 4 bus system suffers the loss of the branch from bus 1 to bus 2. The diagonals on rows 1 and 2 would need to be reduced by 1 while the off diagonals (1,2) and (2,1) would be set to zero. The updated admittance matrix can be seen in Fig. 2.9.

$$-j\begin{bmatrix} 2 & 0 & -1 & -1 \\ 0 & 1 & -1 & 0 \\ -1 & -1 & 2 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix}$$

Fig. 2.9 Admittance Matrix of the Network Shown in Fig. 2.7 After Suffering the Loss of Branch 1 to 2

The OLR algorithm needs to identify areas with the potential to island based on the criteria that a bus or (number of buses) is (are) now connected to the rest of the system by only a single branch as a direct result of the initial outage. For example, for an initial outage of the line from bus 1 to bus 2 in Fig. 2.7, the algorithm must identify any lines that are required to allow buses 1 and 2 to remain connected together. For the example of the 4 bus system, these lines are 2 to 3 and 3 to 1.

As discussed previously, the outage of line 1 to 2 can only create new critical branches between buses 1 and 2. Therefore, power injections of +1 pu at bus 1 and -1pu

at bus 2 are added. All other buses have no power injection. These power injections will establish a gradient in the voltage phase angles throughout the system. The highest angle in the system will be at bus 1 and the lowest at bus 2. The phase angles at the buses that span buses 1 and 2 will lie between the two extremes. Critical branches in the system can be identified by using this phase angle gradient. Recall that the power flowing across a line in the DC power flow is represented as shown in (1.3). With these power injections in the system, if a particular line is required to allow buses 1 and 2 to remain connected together, the power flowing across that line must be 1 pu. Recall that when the OLR was initialized, the impedance of each line was set to $0+j1$ pu. Therefore, the voltage angle difference across any system branch that is required to allow buses 1 and 2 to remain connected together must be equal to ±1 radian. Solving the DC power flow for this 4 bus system yields the voltage phase angles shown in Table 2.2.

Table 2.2 DC Power Flow Solution of the 4 Bus System Example

Bus	1	2	3	4
Voltage Angle	0 rad	-2 rad	-1 rad	0 rad

Notice that the magnitude of the voltage angle difference across the line from bus 2 to bus 3 and the line from bus 3 to bus 1 is 1 radian. This signifies that the lines from bus 2 to bus 3 and bus 3 to bus 1 are both required to allow buses 1 and 2 to remain connected together. Notice also that the magnitude of the voltage angle difference across line 1 to 4 is 0 radian. This signifies that this line is not of interest.

Once the critical branches in the system are found, the algorithm evaluates the number of buses that would be isolated for the loss of each critical branch. This is important to know because an island formation of a single bus may not be of interest to the

system operator. However, if the loss of a single branch would cause the isolation of a significant portion of the system, the operator would need to be alerted. The algorithm determines this information using the voltage phase angle solution of the DC power flow study. It was mentioned previously that there is a gradient in the voltage phase angles throughout the system. For each critical branch the algorithm examines the voltage angles at each end of the line and compares them to the rest of the angles throughout the system. For this example, for the critical line from bus 2 to bus 3 these angles are -2 radians and -1 radian. The algorithm will select the more positive angle. In this case the algorithm will select -1 radian. The algorithm will compare the angle at this bus to the angles at every other bus in the system. All buses that have an angle of -1 radian or higher will remain interconnected following the loss of line from bus 2 to bus 3. For the loss of the branch from bus 2 to bus 3, buses 1, 3, and 4 would remain interconnected. The OLR algorithm can then easily determine the number of buses that would be disconnected following the loss of a particular critical line. Exactly which buses would remain within the island can also be recorded during this process.

A summary of the steps used in the OLR algorithm are given as follows:

1. The first time the OLR is invoked the system admittance matrix is created. All generators, loads, and shunts are neglected and the impedance of each system branch is set to $0+j1$ pu.
2. Following the loss of the system branch from bus A to bus B the system admittance matrix is updated to reflect the loss.

3. A DC power flow is formulated and solved with a power injection of +1 pu at bus A and a power injection of -1 pu at bus B. All other buses have no power injection.
4. The magnitude of the voltage phase angle change across each branch in the system is evaluated. If the magnitude of the voltage phase angle change across any branch in the system is equal to 1 radian then that branch is designated as a critical branch.
5. For each critical branch the algorithm selects the more positive angle on either end of the line. The algorithm compares this angle with the angle at all other buses. Any bus with an angle equal to or larger than this angle will remain together after that critical line is lost. If the number of these buses is X, the algorithm also computes N-X. The lower of these two numbers is reported as the island area.

The OLR algorithm requires factoring the system admittance matrix each time the topology changes and the algorithm is run. Once this is done the DC power flow can be solved and a solution can be obtained. However, the most computationally expensive part of the OLR algorithm is the factoring of the admittance matrix ($O(n)$ to $O(n^3)$). Therefore, the OLR can reduce the time needed to report a solution by using post factoring. Again consider the situation in Fig 2.6. Line h is taken offline and the new critical branches in the system must be found. When the OLR is run, instead of removing line h from the system for the DC power flow it is kept online. This will result in some flow on line h. Assume the flow on line h is 0.8 pu. If the loss of line h creates a singleton cut set in the system, then the cut set in the system with line h online must be of size 2. The sum of the

flows across the cut set must be equal to the power injections. Therefore, if the flow on any line in the system is 0.2 pu, then that line must be part of the cut set including line *h*. This also signifies that this line will become a critical line following the loss of line *h*. The OLR algorithm can apply this information very easily. The target flow that identifies a critical line is normally 1 pu. If the flow on line *h* is *x*, then the new target flow becomes 1-*x*. Using this knowledge the OLR algorithm does not have to factor the system admittance matrix in the first step of the algorithm. Instead the OLR algorithm uses the old topology and uses the factored admittance matrix from the previous run of the OLR algorithm. Once the OLR algorithm reports a solution to the system operator, the admittance matrix can be factored to prepare for the next time it is called. This allows the algorithm to go strait to forward and backward substitution in the DC power flow and allows for a fast solution using $O(E)$ operations.

2.2.2 OLR Algorithm Validation

The OLR algorithm was applied to two different systems for testing purposes. The first of these systems was the 16 bus system shown in Fig. 2.10. This system was chosen to provide examples of OLR solutions that are simple to verify visually. A number of OLR solutions for outages within this system are shown in Table 2.3. Each outage was applied independently of the other outages.

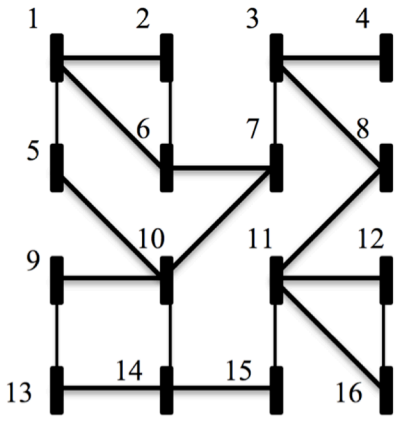

Fig. 2.10 16 Bus Power System Network

The second system used for verification was the IEEE 118 bus system. A diagram of this system can be seen in Fig. 2.11 [46]. This system was used to show the OLR solutions when a system suffers a series of outages that lead to an island formation. This would be very similar to what happened during the Gustav event. A series of outages was applied to this system such that an island would form in the southeast portion of the system. A total of 5 lines were removed in order to form the island. The OLR solution after each outage in the series can be seen in Table 2.3. Notice that after the 4th outage in the series the OLR identifies that the loss of branch 47 to 69 would cause an island formation of 47 buses. Once an outage actually forms an island the OLR can no longer generate a solution.

The OLR algorithm was then applied to the Entergy system to demonstrate the benefit of using the algorithm during a recreation of the Gustav event. The load in the system during hurricane Gustav was very small. As a result, the island would not form before every tie line is lost. If the OLR could detect that only a single tie line remained, that would be a very significant advanced warning that that area of the system needs im-

Table 2.3 OLR Solutions for Several Outages in the 16 Bus System

Initial Outage	Critical Branches	Number of Buses in Identified Area
1 to 6	None	0
6 to 7	1 to 5 5 to 10	3 4
5 to 10	1 to 5 6 to 7	1 4
11 to 15	7 to 3 3 to 8 8 to 11 14 to 15	6 4 3 1
10 to 14	None	0
3 to 8	7 to 3 8 to 11 11 to 15 14 to 15	2 1 4 5

mediate attention. The power flow data for the Entergy system, corresponding to the time hurricane Gustav impacted the system, was provided by Entergy. This power flow case represented the system with just over 20,000 buses. The current implementation of the OLR algorithm does not apply sparsity techniques to reduce the computation requirement when solving large systems. Therefore, the network size had to be reduced in order for the current version of the OLR to be applied to the Entergy system.

Table 2.4 OLR Solutions for a Series of Outages in the IEEE 118 Bus System

Initial Outage	Critical Branches	Number of Buses in Identified Area
65 to 68	None	0
49 to 69	None	0
24 to 72	70 to 71 71 to 72	3 1
70 to 24	23 to 24 47 to 69	1 47
47 to 69	Island forms	Island forms

This reduction could be avoided in practical applications. The island that formed during the Gustav event contained around 140 buses. The OLR algorithm was applied to a 1122 bus portion of the Entergy system network around the Baton Rouge and New Orleans area. A diagram of this section of the Entergy system can be seen in Fig. 2.12. The Matlab code implementation of the OLR algorithm can be seen in Appendix B.

The network of the Entergy system shown in Fig. 2.12 was drawn using PSS/E and does not correspond to the geographical representation of the real Entergy system in this area. The area that disconnected from the rest of the system is illustrated in Fig. 2.12. Recall that 14 tie lines were lost during the hurricane before the island formed. A series of 13 outages were applied to the Entergy system to simulate the progress of the Gustav event. The OLR algorithm was invoked after the loss of each branch. The solutions for all of the tie line outages during the event can be seen in Table 2.5.

The OLR solutions have several interesting aspects. The outage of the first tie line in the series creates two potential islanding areas of 22 and 23 buses. This may have been important to know at the time of the event. Also, the loss of almost every branch creates small potential island areas (all but the 9^{th} outage). After the 13^{th} tie line was lost the OLR algorithm identifies several critical branches that could each potentially create an island of significant size. The actual line outage that created the island was bus 53226 to bus 53173 230 kV. The algorithm identifies that the loss of this line would create an island containing 139 buses. This shows that the OLR algorithm can correctly identify the critical branches in the system. Also, the OLR identifies that there were 5 other critical branches at that operating condition that could have led to a large island formation in the same area.

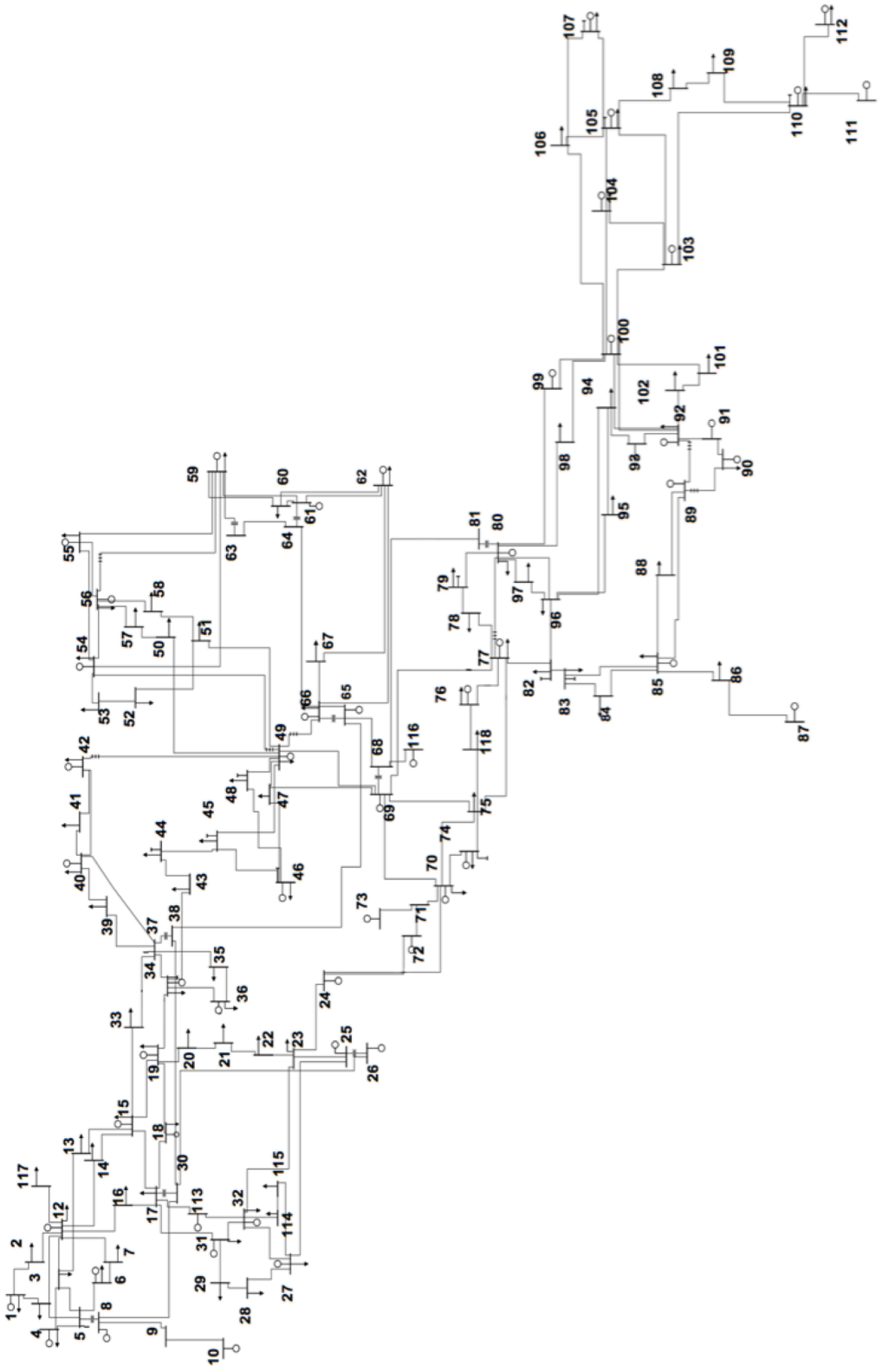

Fig. 2.11 IEEE 118 Bus Power System

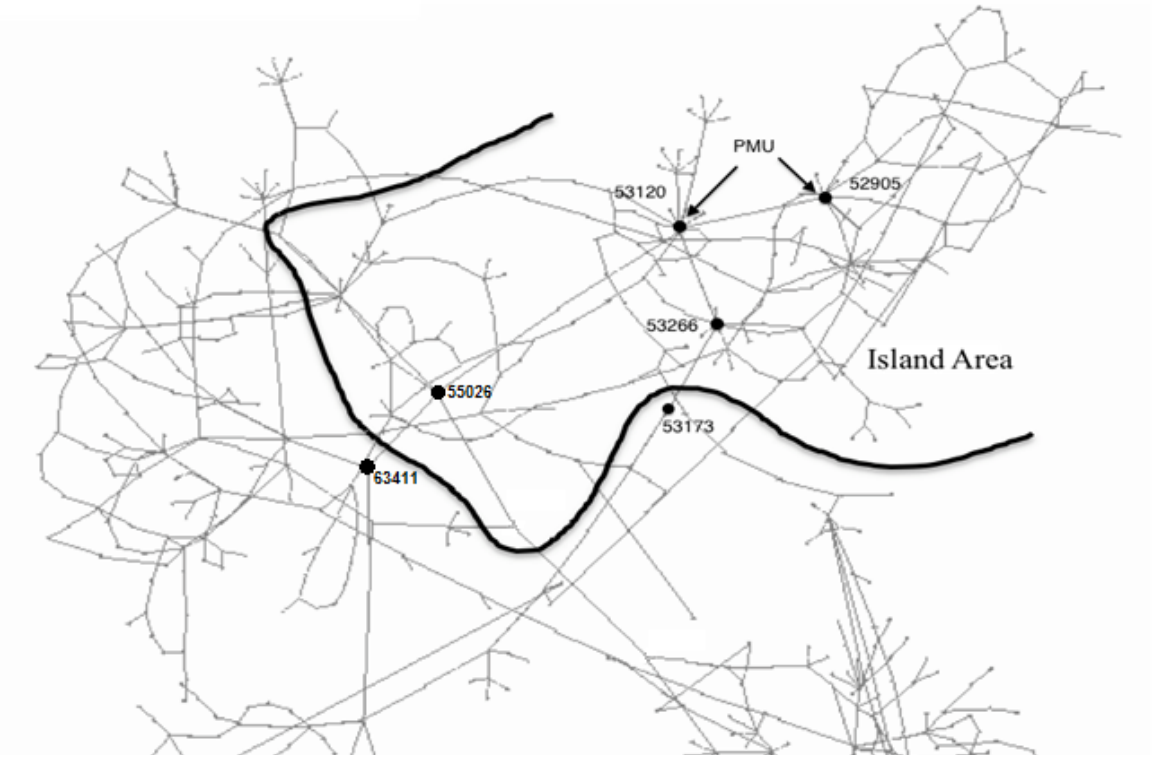

Fig. 2.12 PSS/E Drawing of the Entergy System in Southeast Louisiana. The PMUs Within the Island and the Endpoints of the Final Two Tie Lines are All Labeled.

In the identified area at the time the 13th tie line was lost, there were 3 generators supplying approximately 346 MW. The load in this area was approximately 246 MW with the area exporting 100 MW using the final tie line (the 6 critical branches identified are in series). The goal of the OLR algorithm is to prevent loss of service to customers (if possible) in the event an island forms. If operators had the OLR algorithm output they could have chosen to reduce the flow exiting the island in anticipation of the possible island formation. This would result in higher confidence that the island would remain operational after island formation. Also, if any of the 6 identified branches were reported to have gone offline the operators would know immediately that the island had formed and

Table 2.5 OLR Algorithm Solutions for the Gustav Event

Tie Line Removed	Number of Critical Branches	Number of Buses in Identified Area
1st	3	1, 22, 23
2nd	5	1, 1, 2, 3, 2
3rd	14	1, 4, 3, 1, 3, 11, 13, 11, 8, 9, 6, 10, 10, 2
4th	4	1, 3, 4, 1
5th	6	6, 4, 3, 2, 1, 2
6th	1	1
7th	5	3, 2, 4, 5, 7
8th	8	1, 3, 4, 5, 6, 7, 8, 10
9th	None	0
10th	1	1
11th	4	1, 3, 2, 1
12th	7	6, 7, 5, 1, 1, 2, 3
13th	**53266-53173** 7 others	**139**, 173, 172, 19, 147, 148, 146, 20

required special attention. Recall that in the real event it took 20 minutes to confirm the island had formed.

Together, the network flow and OLR algorithms track changes in the system topology and present results to system operators that have a clear physical significance. Significant power system events sometimes require several major component outages before a rapid system collapse or serious problem occurs [1, 12, 41]. The components that lead to these scenarios are serving as interties to different operating areas. The constant monitoring of these cut sets provides system operators with a fast approximation of the location and severity of potential problems within the power system.

3. LINE OUTAGE DETECTION

3.1 PMU-Based Line Outage Detection

The topology monitoring algorithms presented in the pervious chapter require accurate real-time system topology information in order to be effective. The statuses of components in a power system are usually reported by the SCADA system. SCADA telemetry is then used by the topology processor to determine the structure of the system in real-time [42, 47]. However, during severe system events such as hurricanes, earthquakes, or even system malfunction, some or all SCADA measurements may be unavailable, leading to an uncertain system topology. Prior to such an event, the topology of the system would be known. If changes in line status could be tracked independently of SCADA measurements, an accurate system topology could be maintained. This thesis presents an advanced method of detecting line outages using PMU measurements to be used in conjunction with existing line outage matching techniques [6, 31-37]. The proposed method of line outage detection is accomplished by training a DT to detect the signature of a line outage in PMU measurements.

3.1.1 Hurricane Isaac Simulations

It was discussed in Chapter 1 that a DT method needs sufficient training data in order to be effective. A method that requires having historical events to train with would not be as useful as a method that only requires simulating new events. For this reason, the DT is trained using simulation data. The event in the PMU data available from the Entergy system was withheld from the training data to be used for method validation. To be

sure the trained DT can be applied to the real data for validation, line outages were simulated in the same area as the real event. Specifically, 50 Gustav island formations were created in simulation to supply the line outages needed for DT training. These simulations were conducted using an operations planning representation of the system. It was desired to simulate the line outages using power flow cases with a variety of operating conditions. For this purpose, five power flow cases were provided by Entergy, which corresponded to hurricane Isaac. Hurricane Isaac made landfall at 7:00 PM on August 28, 2012 near the mouth of the Mississippi River [48]. The five power flow cases correspond to different times during the operating horizon; 10:30 AM August 28th, 12:00 PM August 28th, 12:00 PM August 29th, 6:00 PM August 29th, and 12:00 PM August 31st. Along with using the five different operating conditions, ten different orders were used to create a total of 50 simulations. The order of line outages that actually occurred during the Gustav event was included as one of the ten orders. The remaining nine orders were random. The simulations were conducted using PSS/E v33.3. As stated previously, the same island in each simulation was created using one of the five power flow cases and one of the ten tie-line outage orders, providing a total of 50 simulations. During each simulation, the values of bus frequency, voltage magnitude, and voltage phase angle, were recorded at each of the PMU sites at Buses 75400, 76821, 52905, and 53120. Buses 75400 and 76821 are located outside the island area. Buses 52905 and 53120 are located within the island. In each simulation, the 14 tie lines were removed at a rate of one every five seconds until the island formed.

After studying the simulation results, several characteristics were observed. The frequency data in each simulation did not seem to show any useful information. Both the

voltage magnitude and voltage phase angle showed unique characteristics, and the simulations appeared to create two categories. In the first category, a sudden change in difference in phase angle of at least 5º was observed during the removal of at least one tie line. In some simulations, a sudden change in phase angle could be observed for the loss of several lines. A clear example of this can be seen in Fig. 3.1. In the other category, a sudden change in voltage magnitude in the islanded area of at least 5% was observed during the removal of one tie line. An example of this can be seen in Fig. 3.2.

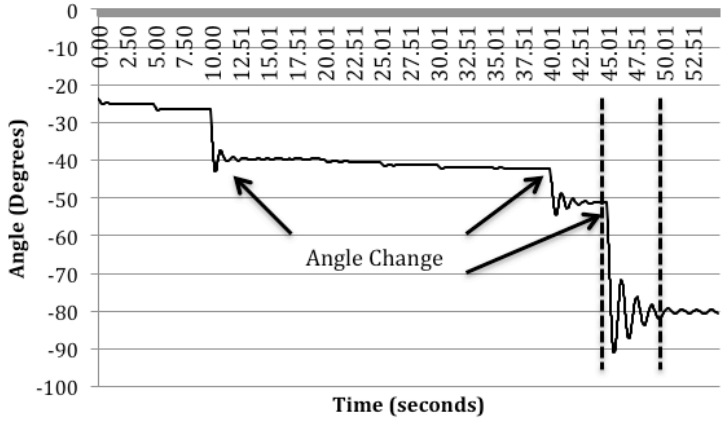

Fig. 3.1 Simulated Voltage Phase Angle Difference Between Buses 75400 and 52905

After looking through the simulations it was found that all 50 simulations fell into category one while 31 of the 50 simulations fell into category two. After studying the simulations, the real PMU voltage magnitude and voltage phase angle data were studied at the times when tie lines were removed from the system. In the real event, last two lines to go offline before the island formed were bus 55026 to bus 63411 500 kV (penultimate) and bus 53266 to bus 53173 230 kV (last).

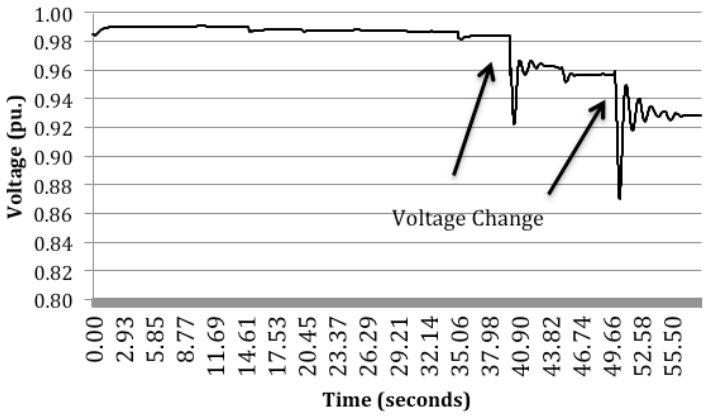

Fig. 3.2 Simulated Voltage Magnitude at Waterford Inside the Island Area

The period of interest was the time when the penultimate tie line went offline. The PMU data at this point showed a signature in the phase angle measurements, but nothing was found in the voltage measurements. A plot of the voltage phase angle difference between the inside and outside of the island can be seen in Fig. 3.3.

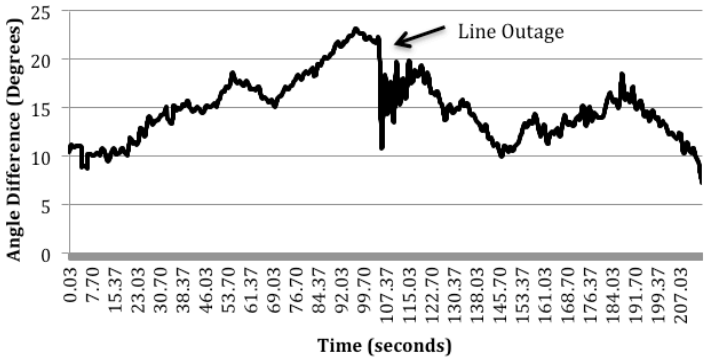

Fig. 3.3 Voltage Phase Angle Difference Between the Inside and Outside of the Island Area Measured by PMUs During the Loss of 55026-63411 (500 kV)

At the moment the 500 kV line was lost, the PMU data observed a ~12º change in the difference between phase angles that stands out from the rest of the data. This is consistent with what was seen in the simulated line outages.

3.1.2 Feature Selection

It was decided to use a feature selection strategy similar to the one used in [39] and [40]. Here, intervals were selected around the times of interest. Rather than using the time series data as the feature to classify, only specific characteristics within each interval are included. Recall that [39] uses a discrete wavelet transform on each interval to determine the wavelet coefficients to identify an island formation around a distributed generator. These coefficients are then used as features in the CART datasets. In the other example, [40] uses root means square and harmonic content of each interval to detect high impedance faults. For this work the features need to capture a sudden large change followed by a smaller oscillation. The features for each interval were then chosen to be:

- End: The value of the last data point in the interval.
- Max1: The largest value in the interval.
- Min1: After the time of Max1, the smallest value in the remaining interval.
- Max2: After the time of Min1, the largest value in the remaining interval.
- Min2: After the time of Max2, the smallest value in the remaining interval.
- Slope 5: The largest slope between any two data points that are 5 data points apart in the interval.
- Slope 10: The largest slope between any two data points that are 10 points apart in the interval.
- Slope 50: The largest slope between any two data points that are 50 points apart in the interval.

The CART database was built using only voltage phase angle data. The intervals selected from the simulation data would begin at the loss of a tie line when a sudden drop in phase angle was observed and would extend approximately 1200 measurements. An example of an interval used in the CART database can be seen between the dotted lines in Fig. 3.1. Before features could be computed the data within the selected interval would need to be preconditioned. It is desired that the DT only look at the relative change in phase angle within the interval. For example, if the phase angle begins at -30º and increases to -20º, then this should be the same as going from 0º to 10º. This was done so that the DT would not need as much training data to define all possible changes in angle. Also, the DT needs to treat positive changes in angle the same as negative changes, for the same reason. For each interval selected, the first data point was defined to be zero and all other data points would be adjusted accordingly. Also, the absolute value of every measurement was taken. Once each interval was selected, the features could be computed. Of the 50 simulations conducted, 80 intervals were selected out of the simulated phase angle data to be used in the CART database. These intervals would have a Class of 1 to indicate a line was lost. The intervals that would have a Class of 0 (or no line lost) were drawn from the real PMU data around times were it was known that no line was tripped. There were 79 intervals of Class 0 selected from the real PMU data to complete the CART database.

3.1.3 Decision Tree Training and Testing

The CART database contains features from 159 intervals. The CART database was divided into two parts. Data from 81 intervals would serve as the learning dataset and

data from 78 intervals would serve as the test dataset. A DT was trained using the learning dataset with equal misclassification cost. The resulting DT can be seen in Fig. 3.4.

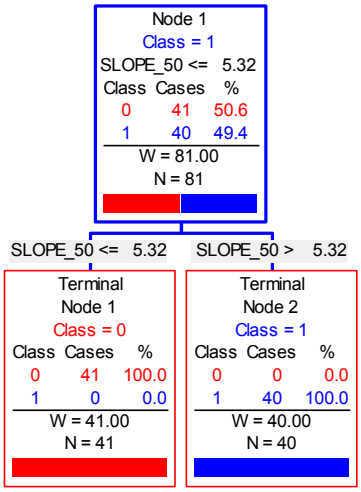

Fig. 3.4 DT Trained with the Learning Dataset

The DT produced by CART contains 2 terminal nodes and has no misclassifications. It states that if the Slope 50 is less than or equal to 5.32 then no line was lost during that time interval. Otherwise, a line was lost. The DT has only one splitting rule using the feature Slope 50. This indicates that the feature Slope 50 is the only feature necessary to determine if a line has tripped. The DT was then tested using the test dataset. The results of the test can be seen in Table 3.1. In Table 3.1, the column "Total Class" lists the total number of samples in the data that should be classified as the "Actual Class." The possible classes are 1 indicating an outage, and 0 indicating no outage. The column labeled "Class 0" indicates how many times the DT classified the inputs from the pool in "Total Class" as 0. The column "Percent Correct" for row one is found by dividing column "Class 0" by column "Total Class." The test database contains data from 39 intervals that should be classified as 0, and 39 intervals that should be classified as 1. The

DT was able to correctly classify every case in the test database, giving the DT 100% accuracy in this test.

Table 3.1 DT Training Results

Actual Class	Total Class	Percent Correct	Class 0	Class 1
0	39	100%	39	0
1	39	100%	39	0

This DT was also applied to the real PMU voltage phase angle data around the time that line 55026 to 63411 was reported to have gone offline. The DT applied to 45 minutes of PMU voltage phase angle data can be seen in Fig. 3.5. The features in this figure were caluclated using a sliding interval. Each time the interval advances, the features for that interval would be calculated and dropped down the DT.

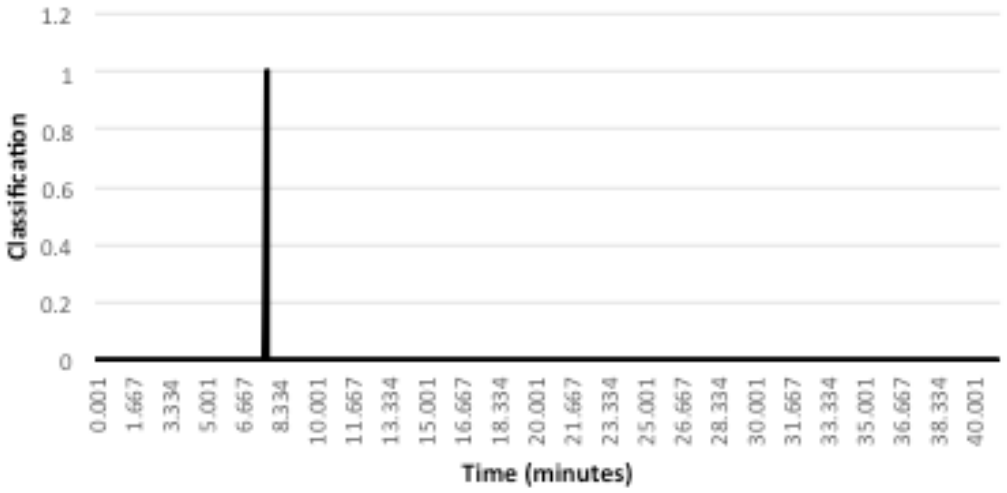

Fig. 3.5 DT Applied to PMU Measurements of the Real 500 kV Line Outage

The DT made several classifications of 1 near minute 8. This is the same time that line 55026 to 63411 was known to have gone offline. The DT also classified all other samples as 0. These results indicate that this method of line outage detection provides a

reliable way of detecting changes in system topology. The DT was trained with only simulated line outages but still had great performance when tested using real PMU measurements. This method can be used in conjunction with existing line outage matching techniques to maintain an acurate system topology in real-time in the event of SCADA data telemetry deficiencies. This method can also be used as a way to verify SCADA line outage reports. If system topology informaiton can be maintained, the topology monitoring methodology proposed in this work can be fully utilized during system disturbances. This method of PMU line outage detection using decision trees is included in the publication [49].

4. CONCLUSIONS

This dissertation proposes a real-time topology monitoring scheme for power systems intended to provide enhanced situational awareness during major system disturbances. The topology monitoring scheme requires accurate real-time topology information to be effective. This scheme is supported by advances in transmission line outage detection based on data-mining PMU measurements.

A network flow analysis is capable of tracking the change in user defined minimal cut sets within a power system, which can alert system operators to significant changes in system topology. The network flow approach proposed in this work makes use of well known graph theory network flow algorithms, but also introduces a new algorithm for updating an old network flow solution for the loss of only a single system branch. The network flow approach was applied to the Entergy system in a recreation of the Gustav island formation. The algorithm was able to track significant changes in the system topology and visually present clear indication that a portion of the system was being severally effected by transmission line outages. For cases within the 20,000-bus Entergy system, the Matlab implementation of Edmonds and Karp requires approximately 0.6 s to generate an initial solution. However, updating a previous solution for the loss of a type one or type two edge requires 0.06 s. Updating for the loss of a type zero edge requires only 0.002 s. Also, because power systems tend to be extremely sparse, the vast majority of branches in a given solution do not contain flow. Therefore, the proposed method of updating pervious network flow solutions provides a significant computational advantage in this application.

Network flow based topology monitoring provides unique benefits to system operations. The algorithm only requires network topology information in order to generate a solution. This approach can supplement conventional system monitoring methods and serve as backup monitoring in the event of computer system failures. The proposed method is also fast enough to be used with a large system model. This allows system operations to monitor the topology of a large area beyond the network managed by state estimation and real-time contingency analysis.

This work also presents a method of determining all singleton cut sets within a given network topology called the OLR algorithm. The OLR can provide additional information in the event a minimal cut set is determined to be of size one. The OLR can also identify less significant potential island formations that are not visible in the user defined minimal cut sets being monitored. The OLR algorithm activates after the loss of a transmission line and determines if any singleton cut sets were created. The algorithm also identifies all groups of buses that would remain connected together following the loss of a singleton cut set. The OLR algorithm requires only standard matrix operations, which are common in power system analysis studies. This allows utilities to easily implement and maintain their own program as opposed to other methods. This method was applied to a recreation of the Gustav island event. The algorithm correctly identified the last tie line lost before the island of 139 buses was formed as well as 5 other lines that could have created an island of similar size and location.

The topology analysis algorithms proposed in this work are supported by line outage detection using PMU measurements aimed at providing accurate real-time topology information. This process uses a DT based data-mining approach to characterize a lost tie

line in simulation. A DT was trained using intervals containing simulated line outages and intervals of real PMU data that did not contain an event. The DT training determined Slope 50 was the best and only feature necessary to detect the outage of transmission lines. The threshold for this feature to detect a line outage was found to be Slope 50 > 5.32. The trained DT was applied to real PMU measurements around the time a 500 kV line was known to have gone offline. The DT was able to correctly identify the time of the 500 kV line outage with no misclassifications. This method of DT training is an accurate and effective method of detecting line outages in PMU measurements even when large variations exist within the data.

This method of line outage detection provides a reliable way of detecting changes in system topology. This method can be used in conjunction with existing line outage matching techniques to maintain an acurate system topology in real-time in the event of SCADA data telemetry deficiencies. If system topology informaiton can be maintained, the topology monitoring methodology proposed in this work can provide insight into potential problems in the network even in the absence of state estimation and real-time contingency analysis.

The topology monitoring method proposed in this work is one implementation of local vulnerability metrics using minimum cut sets. This work could be extended to includeed more complex setups. For example, every line within a power system does not need to be included in the study when applying the method. By only including specific elements, the method could be tuned for a specific system. One possible extension of this work is determining the optimal setup when using network flows in a specific system. Also, the maximum flows found in this work focus on the special case when all edges have a

capacity of one unit of flow. Another possible extension of this work is using maximum flows when edges have different capacities such as the maximum power flow across each line or the present value of power flowing. Including additional studies using different edge capacities in conjunction with the strategy proposed in this work may result in improved performance.

The greatest difficulty faced when using PMU line outage detection in a real power system is the fact that power systems today contain very few PMUs compared to the number of buses in each system. When only a small number of PMUs exist in a power system, not all line outages will be observable at all operating conditions. Additional work could be done by developing a method to find all observable line outages within a power system for the present system operating condition. This would allow operators to know which line outages could be tracked using a PMU measurements.

REFERENCES

[1] FERC Staff, “The Con Edison Power Failure of July 13 and 14, 1977,” *U.S. Department of Energy Federal Energy Regulatory Commission,* pp. 2, 3, 19-20, June 1978.

[2] Bollobás, B., *Modern Graph Theory*. New York, NY: Springer, pp. 8, 9, 67-73, 1998.

[3] Kolluri, S.; Mandal, S.; Galvan, F.; and Thomas, M., "Island Formation in Entergy Power Grid during Hurricane Gustav," in *Power & Energy Society General Meeting,* pp. 1-5, 2009.

[4] H. S. Wilf, *Algorithms and Complexity*, Philadelphia, PA: University of Pennsylvania, pp. 63-77, 1994.

[5] A. V. Goldberg; E. Tardos; R. E. Tarjan, *Network Flow Algorithms*, New York, NY: Springer, pp. 105, 106, 114-119, 1990.

[6] Tate, J.E.; Overbye, T.J., "Line Outage Detection Using Phasor Angle Measurements," in *IEEE Transactions on Power Systems*, vol. 23, no. 4, pp. 1644-1652, November 2008.

[7] De La Ree, J.; Centeno, V.; Thorp, J.S.; Phadke, A.G., "Synchronized Phasor Measurement Applications in Power Systems," in *IEEE Transactions on Smart Grid*, vol. 1, no. 1, pp. 20-27, June 2010.

[8] Cokkinides, G.J.; Meliopoulos, A.P.S.; Stefopoulos, G.; Alaileh, R.; Mohan, A., "Visualization and Characterization of Stability Swings via GPS-Synchronized Data," in *40th Annual Hawaii International Conference on System Sciences*, pp.120, January 2007.

[9] Phadke, A.G., "Synchronized Phasor Measurements in Power Systems," in *IEEE Computer Applications in Power*, vol. 6, no. 2, pp. 10-15, April 1993.

[10] Entergy, “EntergyFACTS,” [Online] Available: http://entergy.com/about_entergy/entergy_facts.aspx

[11] Kolluri, S.; Mandal, S.; Galvan, F.; and Thomas, M., "Island Formation in Entergy Power Grid during Hurricane Gustav," in *IEEE Power & Energy Society General Meeting*, pp. 1-5, 2009.

[12] Galvan, F.; Mandal, S.; Thomas, M.; "Phasor Measurement Units (PMU) Instrumental in Detecting and Managing the Electrical Island Created in the Aftermath of Hurricane Gustav," in *IEEE Power Systems Conference and Exposition*, pp. 1-4. March 2009.

[13] Vittal, V., "An Online Dynamic Security Assessment Scheme using Phasor Measurements and Decision Trees," PSERC seminar, Feb. 2008. [Online]. Available: http://www.pserc.org/cgipserc/getbig/generalinf/presentati/psercsemin1/4psercsemin/vittal_pserc_project_s-27_tele-seminar_slides_021908.pdf

[14] Skiena, S., *The Algorithm Design Manual.* 2nd ed. New York, NY: Spring, pp. 162, 2008.

[15] Wood, A. J.; Wollenberg, B. F., *Power Generation Operation and Control*, 2nd ed. New York, NY: Wiley, pp. 108-111, 1996.

[16] Tinney, W.; Meyer, S.; "Solution of Large Sparse Systems by Ordered Triangular Factorization," in *IEEE Transactions on Automatic Control*, vol. ac-18, no. 4, pp. 333-346, August 1973.

[17] Tinney, W.; Walker, J.; "Direct Solutions of Sparse Network Equations by Optimally Ordered Triangular Factorization," in *Proceedings of the IEEE*, vol. 55, no. 11, pp. 1801-1809, November 1967.

[18] Bittencourt, H.R.; Clarke, R.T., "Use of Classification and Regression Trees (CART) to Classify Remotely-Sensed Digital Images," in *IEEE Geoscience and Remote Sensing Symposium*, vol. 6, pp. 3751-3753, July 2003.

[19] IBM, "CART Algorithm," [Online]. Available: ftp://ftp.boulder.ibm.com/software/analytics/spss/support/Stats/Docs/Statistics/Algorithms/14.0/TREE-CART.pdf

[20] Ford, L. R., Jr.; Fulkerson, D. R, "Maximal Flow Through a Network," *Canadian Journal of Mathematics*, 8(1965), pp. 399-404.

[21] Dinic, E. A., "Algorithm for Solution of a Problem of Maximum Flow in Network with Power Estimation," *Soviet Mathematic – Doklady*, 11(1970), pp. 1277-1280.

[22] Boykov, Y.; Kolmogorov, V., "An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision," in *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 26, no. 9, pp. 1124-1137, September 2004.

[23] Rai, S., “A Cutset Approach to Reliability Evaluation in Communication Networks,” *in IEEE Transactions on Reliability*, vol. R-31, no. 5, pp. 428-431, December 1982.

[24] Ariyoshi, H., “Cut-Set Graph and Systematic Generation of Separating Sets,” in *IEEE Transactions on Circuit Theory*, vol. ct-19, no. 3, pp. 233-240, May 1972.

[25] Cheng, L.; Liu, M.; Ye, C.; Li, Q.; Zhao, Q., “An Extended Minimal Cut Set Algorithm Applied into Overall Power System Reliability Assessment,” in *International Conference on Power System Technology*, pp. 1-6, October 2014.

[26] Jia, Y.; Xu, Z., "A Graph-algebraic Approach for Detecting Islands in Power System," in *IEEE Innovative Smart Grid Technologies Europe (ISGT EUROPE),* pp.1-5, October 2013.

[27] Tsai, M., "Development of Islanding Early Warning Mechanism for Power Systems," in *IEEE Power Engineering Society Summer Meeting*, vol. 1, pp. 22-26, 2000.

[28] Theodoro, E.A.R.; Benedito, R.A.S.; Alberto, L.F.C., "A Fast Method for Islanding Analysis in Power System Grids," in *IEEE International Symposium on Circuits and Systems (ISCAS),* pp.1856-1859, May 2011.

[29] Tarjan, R. E., “A Note on Finding the Bridges of a Graph,” *Information Processing Letters,* vol. 2, no. 6, pp. 160-161, 1974.

[30] Güler, T.; Gross, G., "Detection of Island Formation and Identification of Causal Factors Under Multiple Line Outages," in *IEEE Transactions on Power Systems*, vol. 22, no. 2, pp. 505-513, May 2007.

[31] Tate, J.E.; Overbye, T.J., "Double Line Outage Detection Using Phasor Angle Measurements,” in *IEEE Power & Energy Society General Meeting*, pp.1-5, July 2009.

[32] Abdelaziz, A.Y.; Mekhamer, S.F.; Ezzat, M.; El-Saadany, E.F., "Line Outage Detection Using Support Vector Machine (SVM) based on the Phasor Measurement Units (PMUs) technology," IEEE Power and Energy Society General Meeting, pp.1-8, July 2012.

[33] Sehwail, H.; Dobson, I., "Locating Line Outages in a Specific Area of a Power System with Synchrophasors," in *North American Power Symposium (NAPS),* pp.1-6, 2012.

[34] Emami, R.; Abur, A., "External System Line Outage Identification Using Phasor Measurement Units," in IEEE *Transactions on Power Systems,* vol. 28, no. 2, pp. 1035-1040, May 2013.

[35] Zhao, L.; Song, W.; Tong, L.; Wu, L., "Monitoring for Power-Line Change and Outage Detection in Smart Grid via the Alternating Direction Method of Multipliers," in 28th *International Conference in Workshops on Advanced Information Networking and Applications (WAINA),* pp. 342-346, May 2014.

[36] Chen, J.; Li, W.; Wen, C.; Teng, J.; Ting, P., "Efficient Identification Method for Power Line Outages in the Smart Power Grid," in *IEEE Transactions on Power Systems*, vol. 29, no. 4, pp. 1788-1800, July 2014.

[37] He, M.; Zhang, J., "Fault Detection and Localization in Smart Grid: A Probabilistic Dependence Graph Approach," *First IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pp. 43-48, 2010.

[38] Chen, C.; Hsu, C; Chiu, H.; Rau, H., "Prediction of Survival in Patients with Liver Cancer using Artificial Neural Networks and Classification and Regression Trees," in *Seventh International Conference on Natural Computation*, pp. 811-815, July 2011.

[39] Lidula, N.W.A.; Rajapakse, A.D., "A Pattern Recognition Approach for Detecting Power Islands Using Transient Signals—Part I: Design and Implementation," in *IEEE Transactions on Power Delivery*, vol. 25, no. 4, pp. 3070-3077, October 2010.

[40] Sheng, Y.; Rovnyak, S.M., "Decision Tree-Based Methodology for High Impedance Fault Detection," in *IEEE Transactions on Power Delivery*, vol. 19, no. 2, pp. 533-536, April 2004.

[41] Werho, T.; Vittal, V.; Kolluri, S.; Wong, S. M., "A Potential Island Formation Identification Scheme Supported by PMU Measurements," in *IEEE Transactions on Power Systems*, vol. PP, no. 99, pp. 1-9, February 2015.

[42] Dondeti, J. R.; Yang, C.; Trotter, K.; Witmeier, A; Sherd, K., "Experiences with Contingency Analysis in Reliability and Market Operations at MISO," in *IEEE Power and Energy Society General Meeting*, pp. 1-7, July 2012.

[43] Davis, M. C.; Overbye, T. J., "Multiple Element Contingency Screening," in *IEEE Transactions on Power Systems*, vol. 26, no. 3, pp. 1294-1301, August 2011.

[44] NERC Steering Group, "Technical Analysis of the August 14, 2003, Blackout: *What happened, Why, and What Did We Learn?" North American Electric Reliability Council*, pp. 1-5, 27-52, July 13th, 2004.

[45] FERC and NERC Staff, "Arizona-Southern California Outages on September 8, 2011," *Federal Energy Regulatory Commission and the North American Electric Reliability Corporation,* pp. 23-47, 86-88, April 2012.

[46] Christie, R., IEEE Power System Test Cases from EE of UW. University of Washington., Washington. [Online]. Available: http://sys.elec.kitami-it.ac.jp/ueda/demo/WebPF/testdata.html

[47] Huang, J. A; Loud, L.; Vanier, G.; Lambert, B.; Guillon, S., "Experiences and Challenges in Contingency Analysis at Hydro-Quebec," in *IEEE Power and Energy Society General Meeting*, pp. 1-9, July 2012.

[48] Jervis, R., "Hurricane Isaac Pounds Louisiana, Water Pours Over Levee," USA Today, August 29th, 2012.

[49] Werho, T; Vittal, V.; Kolluri, S.; Wong, S. M., "Power System Connectivity Monitoring Using a Graph Theory Network Flow Algorithm," in *IEEE Transactions on Power Systems* – Under review

APPENDIX A

MATLAB IMPLEMENTATION OF NETWORK FLOW ALGORITHMS

```
function[]=Network_Flows

%-------------------------------------------------
%Network Flow Algorithms
%Trevor Werho
%2015
%Matlab (R2011a)

%Determines the minimum cut set between a source and a sink in
%a network using Edmonds and Karp
%After the removal of a branch is specified the solution
%is updated using the old solution
%The process of removing and updating can be repeated
%-------------------------------------------------

clc
format short
%specify the source bus number
source=1;

%specify the sink bus number
sink=2;

%constructs the network and generates the initial solution
[S,Flows,LinesIn,New_num,Vec1,Vec2,N,r_org,c_org]=Points(source,sink);
S

%specifies the branch that is removed and updates the solution
%the first two inputs define the removed branch, example: (3,4,S…
[S,Flows,LinesIn]=lineout(3,4,S,Flows,LinesIn,...
    New_num,Vec1,Vec2,N,r_org,c_org);
S

end

function[S,Flows,LinesIn,New_num,Vec1,Vec2,N,r_org,c_org]=Points(r,c)

%constructs the network and generates the initial solution

[New_num,N,T,F,Status,Vec1,Vec2,LinesIn,Flows]=BuildYbus(r,c);

%generates the initial solution using the Edmonds and Karp method
[S,Flows,LinesIn]=mostflow(r,c,New_num,LinesIn,Flows);

r_org=New_num(r);
c_org=New_num(c);
end

function[New_num,N,T,F,Status,Vec1,Vec2,LinesIn,Flows]=BuildYbus(r,c)

%this function reads the system information in from a text file
%used to initialize the program
%the file is a list of system branches in the form:
%from bus  to bus  resistance  reactance susceptance  line-status
%(1 or 0)
```

```
[T, F, R, X, B, Status]=textread('file name.txt','%d %d %f %f %f %d');
M=length(T);

%the buses are renumber starting at bus 1
check=zeros(1,505000);
New_num=zeros(1,505000);

%the source location is specified to be bus 1
New_num(r)=1;
check(r)=1;

i=1;
k=2;
while i < M+1
    if check(T(i)) == 0
        if Status(i)==1

                if T(i) ~= c
            New_num(T(i))=k;
            check(T(i))=1;
            k=k+1;
                end

        end
    end

    if check(F(i)) == 0
        if Status(i)==1

                if F(i) ~= c
            New_num(F(i))=k;
            check(F(i))=1;
            k=k+1;
                end

        end
    end
    i=i+1;
end

New_num(c)=k;
check(c)=1;

%the sink location is specified to be the last bus
N=k;
Ybus=zeros(N);

Old_num=zeros(1,N);
i=1;
while i < 505000+1

    if New_num(i) ~= 0
    Old_num(New_num(i))=i;
```

```
    end

    i=i+1;
end

%build graph sparse matrix
i=1;
k=1;
while i < M+1

    if check(T(i)) == 1

        if check(F(i)) == 1

            if Status(i) == 1

                  Vec1(2*k-1)=New_num(T(i));
                  Vec1(2*k)=New_num(F(i));
                  Vec2(2*k-1)=New_num(F(i));
                  Vec2(2*k)=New_num(T(i));
                  k=k+1;

             end
        end
    end
    i=i+1;
end

M=k-1;

LinesIn=sparse(N,N); %keeps track of all lines that are in service
Flows=sparse(N,N); %keeps track of all lines that contain flow

%records the lines that are in service in a sparse matrix
M=length(Vec1);

i=1;
while i < M+1

    LinesIn(Vec1(i),Vec2(i))=1;

    i=i+1;
end

end

function[S,Flows,LinesIn]=lineout(r,c,S,Flows,LinesIn,New_num,...
    Vec1,Vec2,N,r_org,c_org)

%this function updates a network flow solution following the loss
%of a system branch


%checks for line flow
```

```
step=3;

if Flows(New_num(r),New_num(c))==0
    if Flows(New_num(c),New_num(r))==0
    LinesIn(New_num(r),New_num(c))=0;
    LinesIn(New_num(c),New_num(r))=0;
    step=0;
    end
end

if Flows(New_num(r),New_num(c))==1
    if Flows(New_num(c),New_num(r))==0
    Flows(New_num(r),New_num(c))=0;
    LinesIn(New_num(r),New_num(c))=0;
    LinesIn(New_num(c),New_num(r))=0;
    step=1;
    end
end

if step==3
if Flows(New_num(r),New_num(c))==0
    if Flows(New_num(c),New_num(r))==1
    Flows(New_num(c),New_num(r))=0;
    LinesIn(New_num(r),New_num(c))=0;
    LinesIn(New_num(c),New_num(r))=0;
    step=2;
    end
end
end

%for a flow from r to c
if step == 1

    %check for augmenting path
    [D,path]=graphshortestpath(LinesIn,New_num(r),...
    New_num(c),'Method','BFS','Directed','true');

    %if a path is found the edge is type 2
    if D < 999999999

        i=1;
        while i < D+1

            LinesIn(path(i),path(i+1))=0;
            Flows(path(i),path(i+1))=1;


            if Flows(path(i+1),path(i))==1
                 Flows(path(i),path(i+1))=0;
                 Flows(path(i+1),path(i))=0;
            end

            i=i+1;
        end
```

```
    end

    %if no path is found the edge is type 1
    if D > 999999999

        if r_org ~= New_num(r)
        [D,path]=graphshortestpath(Flows,r_org,...
            New_num(r),'Method','BFS','Directed','true');

        i=1;
        while i < D+1

            Flows(path(i),path(i+1))=0;
            LinesIn(path(i),path(i+1))=1;

            i=i+1;
        end

        end

        if c_org ~= New_num(c)
        [D,path]=graphshortestpath(Flows,New_num(c),...
            c_org,'Method','BFS','Directed','true');

        i=1;
        while i < D+1

            Flows(path(i),path(i+1))=0;
            LinesIn(path(i),path(i+1))=1;

            i=i+1;
        end

        end

        S=S-1;

    end
end

%for a flow from c to r
if step == 2

    %if a path is found the edge is type 2
    [D,path]=graphshortestpath(LinesIn,New_num(c),...
        New_num(r),'Method','BFS','Directed','true');

    if D < 999999999

        i=1;
        while i < D+1

            LinesIn(path(i),path(i+1))=0;
            Flows(path(i),path(i+1))=1;
```

```
            if Flows(path(i+1),path(i))==1
                 Flows(path(i),path(i+1))=0;
                 Flows(path(i+1),path(i))=0;
            end

            i=i+1;
        end

    end

    %if no path is found the edge is type 1
    if D > 999999999

        if r_org ~= New_num(r)
        [D,path]=graphshortestpath(Flows,r_org,New_num(c),...
            'Method','BFS','Directed','true');

        i=1;
        while i < D+1

            Flows(path(i),path(i+1))=0;
            LinesIn(path(i),path(i+1))=1;

            i=i+1;
        end

        end

        if c_org ~= New_num(c)
        [D,path]=graphshortestpath(Flows,New_num(r),c_org,...
            'Method','BFS','Directed','true');

        i=1;
        while i < D+1

            Flows(path(i),path(i+1))=0;
            LinesIn(path(i),path(i+1))=1;

            i=i+1;
        end

        end

        S=S-1;

    end

end

end
```

```
function[S,Flows,LinesIn]=mostflow(r,c,New_num,LinesIn,Flows)

%this function determines the initial network flow solution using
%Edmonds and Karp

count=0;
flag=0;
while flag < 1

    [S,path]=graphshortestpath(LinesIn,New_num(r),...
        New_num(c),'Method','BFS','Directed','true');


if S < 999999999
    count=count+1;
i=1;
while i < S+1

    LinesIn(path(i),path(i+1))=0;
    Flows(path(i),path(i+1))=1;

    i=i+1;
end
else
    flag=1;
end
end
S=count;
end
```

APPENDIX B

MATLAB IMPLEMENATION OF THE OLR ALGORITHM

```
function[]=OLR_Algorithm

%-----------------------------------------------------
%OLR Algorithm
%Trevor Werho
%2014
%Matlab (R2011a)

%Determines all singleton cut sets in a system following the removal of
%a specified line
%-----------------------------------------------------

clc

%command to build the admittance matrix for the system
%only used once to initialize the program
%[Ybus,New_num,N,T,F,Status]=BuildYbus();
[Ybus,New_num,N,T,F,Status]=BuildYbus();

%Define the line that has gone offline
%update the admittance matrix to reflect the loss
%[Ybus,r,c]=ModYbus(Ybus,bus #, bus #,New_num);
[Ybus,r,c]=ModYbus(Ybus,1,2,New_num);

%display the first bus number of the outaged line
r

%display the second bus number of the outaged line
c

%Solve the DC power flow for injections at both ends of the
%outages line
[Ang2]=FindAngles(Ybus,r,c,New_num);

%Find all critical lines as well as the area that corresponds to
%each line
[Branches,Area]=FindLines(Ang2,T,F,Status,New_num,N)

end

function[Ybus,New_num,N,T,F,Status]=BuildYbus()

%this function reads the system information in from a text file
%used to initialize the program
%the file is a list of system branches in the form:
%from bus  to bus  resistance  reactance susceptance  line status
%(1 or 0)
[T, F, R, X, B, Status]=textread('file name.txt','%d %d %f %f %f %d');
M=length(T);
check=zeros(1,505000);
New_num=zeros(1,505000);

%the buses are renumbered starting at bus 1
i=1;
k=1;
```

```
while i < M+1
    if check(T(i)) == 0
        if Status(i)==1
            New_num(T(i))=k;
            check(T(i))=1;
            k=k+1;
        end
    end

    if check(F(i)) == 0
        if Status(i)==1
            New_num(F(i))=k;
            check(F(i))=1;
            k=k+1;
        end
    end
    i=i+1;
end

%the program checks for double lines
%double lines are represented as one connection
i=1;
tempnum=0;
while i < 505001

    tempnum=tempnum+check(i);

    i=i+1;
end
N=tempnum;
Ybus=zeros(N);

%if Same_X=1 then lines resistances and reactances are ignored
Same_X=1;

%builds the system admittance matrix
i=1;
while i < M+1

    if check(T(i)) == 1

        if check(F(i)) == 1

            if Status(i) == 1

                if Same_X == 0


Ybus(New_num(T(i)),New_num(F(i)))=Ybus(New_num(T(i)),...
                    New_num(F(i)))-1/(R(i)+1i*X(i));

Ybus(New_num(F(i)),New_num(T(i)))=Ybus(New_num(F(i)),...
                    New_num(T(i)))-1/(R(i)+1i*X(i));

Ybus(New_num(T(i)),New_num(T(i)))=Ybus(New_num(T(i)),...
```

```
                    New_num(T(i)))+1/(R(i)+1i*X(i));

Ybus(New_num(F(i)),New_num(F(i)))=Ybus(New_num(F(i)),...
                    New_num(F(i)))+1/(R(i)+1i*X(i));

                else


Ybus(New_num(T(i)),New_num(F(i)))=Ybus(New_num(T(i)),...
                    New_num(F(i)))-1;

Ybus(New_num(F(i)),New_num(T(i)))=Ybus(New_num(F(i)),...
                    New_num(T(i)))-1;

Ybus(New_num(T(i)),New_num(T(i)))=Ybus(New_num(T(i)),...
                    New_num(T(i)))+1;

Ybus(New_num(F(i)),New_num(F(i)))=Ybus(New_num(F(i)),...
                    New_num(F(i)))+1;

                end

            end
        end
    end
    i=i+1;
end

end

function[Ybus,r,c]=ModYbus(Ybus,r,c,New_num)

%this function updates the admittance matrix for the loss of a line

Ybus(New_num(r),New_num(r))=Ybus(New_num(r),New_num(r))+...
    Ybus(New_num(r),New_num(c));
Ybus(New_num(c),New_num(c))=Ybus(New_num(c),New_num(c))+...
    Ybus(New_num(r),New_num(c));
Ybus(New_num(r),New_num(c))=0;
Ybus(New_num(c),New_num(r))=0;

end

function[Ang2]=FindAngles(Ybus,r,c,New_num)

%this function solves the DC power flow

N=length(Ybus);
I=zeros(N,1);
Ang=zeros(N,1);

I(New_num(r))=1;
I(New_num(c))=-1;

Ybus2=Ybus;
```

```
I2=I;
Ang2=Ang;

Ybus2(N,:)=[];
Ybus2(:,N)=[];
I2(N)=[];
Ang2(N)=[];

[Q,R] = qr(Ybus2);
A=Q'*I2;
Ang2=R\A;
Ang2(N)=0;
end

function[Branches,Area]=FindLines(Ang2,T,F,Status,New_num,N)

%this function finds the critical lines and areas

M=length(T);
Branches=[0,0];
Area=[0];

%critical lines are found and stored in Branches
p=1;
i=1;
while i < M+1

    if Status(i) == 1
    Angle=abs(Ang2(New_num(T(i)))-Ang2(New_num(F(i))));
    if Angle > .999
        if Angle < 1.0001
        Branches(p,1)=New_num(T(i));
        Branches(p,2)=New_num(F(i));
        p=p+1;
        end
    end
    end

    i=i+1;
end

%determines the number of buses in each critical area
%stored in Area
i=1;
while i < p

    temp_ang1=Ang2(Branches(i,1));
    temp_ang2=Ang2(Branches(i,2));
    temp_ang=max(temp_ang1,temp_ang2);

    area_count=0;
    k=1;
    while k < N+1

        if Ang2(k) > temp_ang-.1
```

```
            area_count=area_count+1;
        end

        k=k+1;
    end

    temp_area=min(area_count,N-area_count);

    Area(i)=temp_area;

    i=i+1;
end


%buses are returned to their original numbering
i=1;
while i < 3

    k=1;
    while k < p

        j=1;
        while j < 505001

            if Branches(k,i) == New_num(j)
                Branches(k,i)=j;
                j=505001;
            end

            j=j+1;
        end

        k=k+1;
    end

    i=i+1;
end

end
```